

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

(ICAI) INGENIERO ELECTROMECÁNICO

METODOLOGÍA APLICADA A LA CREACIÓN DE UN SISTEMA DE TEST MEDIANTE LIBRERÍAS TCL

Autor: Pablo Calvo Báscones
Director: Jose Sánchez Almagro

Madrid
Mayo2014



AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN ACCESO *ABIERTO* (*RESTRINGIDO*) DE DOCUMENTACIÓN

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Pablo Calvo Báscones, como estudiante de la UNIVERSIDAD PONTIFICIA COMILLAS (COMILLAS), **DECLARA**

que es el titular de los derechos de propiedad intelectual, objeto de la presente cesión, en relación con la obra Proyecto Fin de Grado**: Metodología aplicada a la creación de un sistema de test mediante librerías TCL** [1], que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual como titular único o cotitular de la obra.

En caso de ser cotitular, el autor (firmante) declara asimismo que cuenta con el consentimiento de los restantes titulares para hacer la presente cesión. En caso de previa cesión a terceros de derechos de explotación de la obra, el autor declara que tiene la oportuna autorización de dichos titulares de derechos a los fines de esta cesión o bien que retiene la facultad de ceder estos derechos en la forma prevista en la presente cesión y así lo acredita.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad y hacer posible su utilización de *forma libre y gratuita* (*con las limitaciones que más adelante se detallan*) por todos los usuarios del repositorio y del portal e-ciencia, el autor **CEDE** a la Universidad Pontificia Comillas de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra (a) del apartado siguiente.

3º. Condiciones de la cesión.

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia, el repositorio institucional podrá:

[1] Especificar si es una tesis doctoral, proyecto fin de carrera, proyecto fin de Máster o cualquier otro trabajo que deba ser objeto de evaluación académica



(a) Transformarla para adaptarla a cualquier tecnología susceptible de incorporarla a internet; realizar adaptaciones para hacer posible la utilización de la obra en formatos electrónicos, así como incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato. .

(c) Comunicarla y ponerla a disposición del público a través de un archivo abierto institucional, accesible de modo libre y gratuito a través de internet.[2]

(d) Distribuir copias electrónicas de la obra a los usuarios en un soporte digital. [3]

4º. Derechos del autor.

El autor, en tanto que titular de una obra que cede con carácter no exclusivo a la Universidad por medio de su registro en el Repositorio Institucional tiene derecho a:

a) A que la Universidad identifique claramente su nombre como el autor o propietario de los derechos del documento.

b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

c) Solicitar la retirada de la obra del repositorio por causa justificada. A tal fin deberá ponerse en contacto con el vicerrector/a de investigación (curiarte@rec.upcomillas.es).

d) Autorizar expresamente a COMILLAS para, en su caso, realizar los trámites necesarios para la obtención del ISBN.

---

[2] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría redactado en los siguientes términos:

(c) Comunicarla y ponerla a disposición del público a través de un archivo institucional, accesible de modo restringido, en los términos previstos en el Reglamento del Repositorio Institucional

[3] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría eliminado.



d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

a) Deberes del repositorio Institucional:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.

- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusiva del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.

- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.



b) Derechos que se reserva el Repositorio institucional respecto de las obras en él registradas:

- retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 28. de Mayo de 2014

ACEPTA

Fdo……………………………………………………………………

Proyecto realizado por el alumno/a:

Pablo Calvo Báscones

Fdo: Fecha: 28/05/2014

Autorizada la entrega del proyecto cuya información no es de carácter confidencial

EL DIRECTOR DEL PROYECTO

Jose Sánchez Almagro

Fdo: Fecha:

V^{O} B^{O} del Coordinador de Proyectos

Álvaro Sánchez Miralles

Fdo: Fecha:



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

(ICAI) INGENIERO ELECTROMECÁNICO

METODOLOGÍA APLICADA A LA CREACIÓN DE UN SISTEMA DE TEST MEDIANTE LIBRERÍAS TCL

Autor: Pablo Calvo Báscones
Director: Jose Sánchez Almagro

Madrid
Mayo2014

METODOLOGÍA APLICADA A LA CREACIÓN DE UN SISTEMA DE TEST MEDIANTE LIBRERÍAS TCL

Autor: Pablo Calvo Báscones

Directores: José Sánchez Almagro

Entidad colaboradora: EADS-Astrium Crisa

RESUMEN DEL PROYECTO

1. INTRODUCCIÓN

La creciente complejidad de los proyectos de ingeniería ha fomentado la aparición de nuevas metodologías de trabajo. Los objetivos que se persiguen al implementar una metodología de trabajo robusta y eficiente son los siguientes:

- Una mayor compatibilidad entre los distintos elementos que componen un proyecto.
- Poder evitar problemas ya solventados ocurridos en casos anteriores.
- Aumentar la eficiencia, tanto en el proceso de diseño como en la ejecución.
- Evitar "puntos ciegos" del proyecto que puedan representar una amenaza para la ejecución del mismo.

Para implantar una metodología de trabajo de forma efectiva, se establecerán tres fases principales:

1) Fase de **identificación**: El objetivo de esta fase, será la identificación del tipo de proyecto que será llevado a cabo.
2) Fase de **formación**: En esta fase se facilitará a cada miembro del grupo técnico los recursos necesarios para una correcta ejecución del proyecto.
3) Fase de **consolidación**: Se establece la metodología que mejor se adapte a las necesidades del proyecto.

OBJETIVO: "Conseguir una elevada calidad en la ejecución del proyecto, disminuyendo costes y aumentado la eficiencia de todo el proceso de ejecución".

- METODOLOGÍAS PRESENTES EN ESTE PROYECTO

En el diseño de cada una de las librerías del sistema, se podrán emplear conjuntamente tres tipos de metodologías de software distintas:

1) Metodologías **estructuradas**: Implementadas en aquellas librerías donde se requiere la ejecución de procesos de forma secuencial.
   Ej: Librería de secuencias de test.

2) Metodologías **orientadas a objetos**: Permitirá dividir cada librería de comandado en clases, objetos y métodos independientes.

   Ej: Librería de comandado de instrumentos.

3) Metodologías **orientadas a sistemas en tiempo real:** Permite el uso de interrupciones, hilos, etc., con el fin de poder controlar varios procesos de forma simultánea.

   Ej: Gestión de errores y situaciones de emergencia del sistema.

En el diseño de las secuencias, podrán seguirse multitud de metodologías de verificación y pruebas de test: Pruebas de caja negra, pruebas de compatibilidad, pruebas de funcionalidad, pruebas de estrés, etc.

2. METODOLOGÍA APLICADA A UN CASO REAL

OBJETIVO: Diseño de un sistema de test automático.

FINALIDAD: Verificar el cumplimiento de las especificaciones técnicas de un convertidor CC/CC.

ESPECIFICACIONES DEL SISTEMA:

- **Modo de ejecución: Automático**
- **Modos de trabajo: Secuencia / Depuración**
  - **Secuencia:** Comandado de instrumentos de forma automática.
  - **Depuración:** Comandado de instrumentos de forma manual.
- **Lenguaje de programación:** TCL
- **Librerías:**
  - **driver_gpib:** Librería encargada de gestionar las comunicaciones entre el sistema de control y el resto de instrumentos (Registro de comandos enviados, verificación de conexiones, etc.).
  - **error_handle:** Librería encargada de gestionar las situaciones de emergencia del sistema, salidas de secuencia controladas, "reporting de errores", etc.
  - **driver_gpib_agil34970a:** Librería encargada del comandado del escáner Agilent 34970A.
  - **driver_gpib_hp6653a:** Librería encargada del comandado de la fuente de alimentación HP 6653A.
  - **driver_gpib_hp34401a:** Librería encargada del comandado del multímetro HP 344010A.
  - **driver_gpib_kikuplz150u:** Librería encargada del comandado de la carga dinámica Kikusui PLZ 150u.

- **driver_rs232_tekttps2024:** Librería encargada del comandado del osciloscopio Tektronix PS2024.
- **Clases abstractas:** Librerías que contienen las clases abstractas de los instrumentos de comandado.

ESPECIFICACIONES DE LA PRUEBA DE TEST DEL CONVERTIDOR:

- **Modo de prueba:** Secuencia de test.
- **Tipo de prueba:** Funcional.
- **Instrumentos que participan en la prueba:**
  - Multímetro HP 344010A.
  - Fuente de alimentación HP 6653A.
  - Carga dinámica Kikusui PLZ 150u.
  - Osciloscopio Tektronix PS2024.

- **Fases de la prueba:**
  - **Encendido del convertidor:** Aumento de tensión a la entrada del convertidor hasta alcanzar la tensión nominal de trabajo. En esta etapa de la prueba, se registrará la tensión de encendido del convertidor.

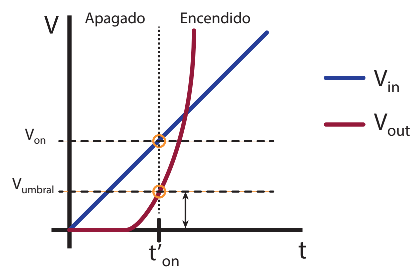


Figura 1. Arranque de un convertidor

  - **Pruebas de eficiencia a tensión constante y carga variable:** Se realizará un estudio sobre el comportamiento del convertidor para distintos valores de carga.
  - **Apagado del convertidor:** Disminución de la tensión de entrada del convertidor hasta a un valor por debajo de la tensión de apagado, indicada en el data-sheet del componente. En esta etapa de la prueba, se registrará la tensión de apagado del convertidor.

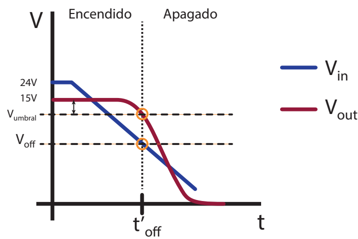


Figura 2. Apagado de un convertidor

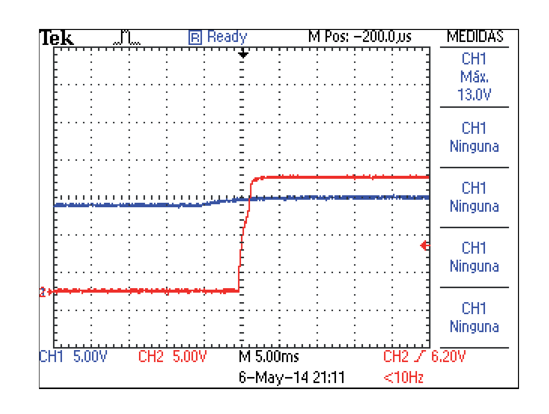
RESULTADOS DE LA PRUEBA:

- **Resultados de la verificación:**

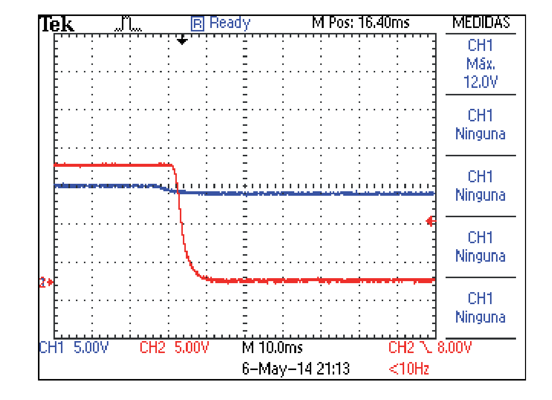
	V_{on} [V]	V_{off} [V]	V_{out} [V]	η_{med}
Espec. Técnicas	12	11	15	84%
Medición	12,17	11,15	15,035	85,25%
Cumple con las especificaciones	✓	✓	✓	✓

Figura 3. Resultados de la prueba.

- **Capturas automáticas del osciloscopio del encendido y apagado del convertidor:**

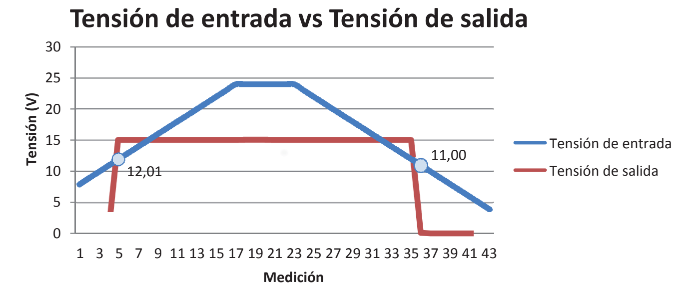


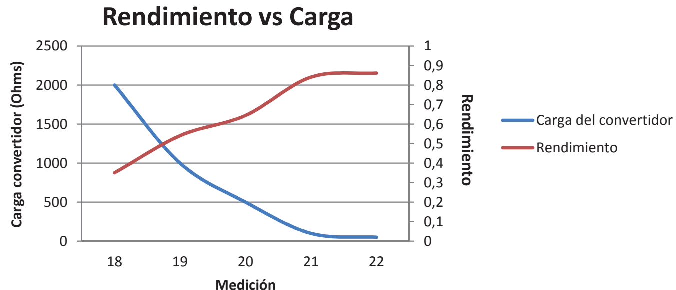
Captura del instante en que se enciende el convertidor.



Captura del instante en que se apaga el convertidor.

- **Análisis del comportamiento del convertidor para distintos valores de carga:**





METHODOLOGY APPLIED IN THE DESIGN OF A TEST SYSTEM BASED ON TCL LIBRARIES

Author: Pablo Calvo Báscones

Director: José Sánchez Almagro

Collaborating institution: EADS-Astrium Crisa

ABSTRACT

1. INTRODUCTION

The rise of the complexity in engineering projects has enhanced the appearance of new working methodologies. The main objectives pursued by implementing a sturdy efficient methodology are the following ones:

- More compatibility between the elements that make up the whole project.
- Avoid any already-solved problem that occurred in previous projects.
- Increase the efficiency, both in the design project and in the execution of it.
- Avoid any project "blind spots" that could become a risk for the execution of it-self.

In order to introduce a working methodology in an effective way, it will be established three main phases:

1) Identification phase: The aim of this phase will be the identification of the type of project that will be executed.
2) Training phase: In this phase, any member of the technical group will be provided with all the training resources needed to get a right execution of the project
3) Consolidation phase: In this last phase, the methodology that better fits the requirements of the project will be established.

OBJECTIVE: "Get a high quality in the execution of the project by reducing costs and increasing the efficiency of the whole execution process"

- METHOLOGIES APPLIED IN THIS PROJECT

Three different types of software methodologies could be use jointly in the design of each library of the system:

1) **Structured** methodologies: These types of methodologies are introduced in those libraries where a sequential execution of the process is required.
   Ex: Test sequence libraries.

2) **Object oriented** methodologies: They make possible to divide each library into independent classes, objects and methods.
   - Ex: Instruments command libraries.
3) **Real time system oriented** methodologies: They make possible to use interruptions, threads, etc. During the sequence, in order to control several process simultaneously.
   - Ex: Libraries in charge of managing all error and emergency situations occurred during the sequence.

During the design of a test sequence, several types of verification methodologies can be introduced: Black box testing, compatibility testing, functional testing, stress/endurance testing, etc.

2. METHODOLOGY APPLIED IN TO A REAL CASE:

OBJETIVE: Design an automated test system.

PURPOSE: Verify the technical specifications of a DC/DC converter.

SYSTEM SPECIFICATIONS:

- **Execution mode**: Automatic
- **Working mode:** Sequence / Debugging
  - **Sequence**: Instrument command in an automatic way.
  - **Debugging**: Instrument command in a manual way.
- **Programming language**: TCL
- **Libraries:**
  - **driver_gpib:** Library that manages the communications between the control system and the rest of the instruments (Command sent register, connection verification, etc.)
  - **error_handle:** Library that manages all emergency situations of the system, the sequence controlled exit, error reporting, etc.
  - **driver_gpib_agil34970a:** Library used to command the scanner: Agilent 34970A .
  - **driver_gpib_hp6653a:** Library used to command the power supply: HP 6653A.
  - **driver_gpib_hp34401a:** Library used to command the multimeter: HP 344010A.
  - **driver_gpib_kikuplz150u:** Library used to command the electronic load: Kikusui PLZ 150u.

- **driver_rs232_tekttps2024:** Library used to command the oscilloscope: Tektronix PS2024.
- **Abstract classes:** Libraries that contain the abstract classes of the commanded instruments.

ESPECIFICACIONES DE LA PRUEBA DE TEST DEL CONVERTIDOR:

- **Test mode:** Sequence.
- **Type of testing:** Functional.
- **Instruments that take part in the sequence:**
  - Multimeter: HP 344010A.
  - Power supply HP 6653A.
  - Electronic load Kikusui PLZ 150u.
  - Oscilloscope Tektronix PS2024.

- **Stage of the test sequence:**
  - **Electronic ignition of the converter :** The input voltage of the converter will increase until reaching the nominal operating voltage. In this step of the sequence, the converter ignition voltage level will be registered.

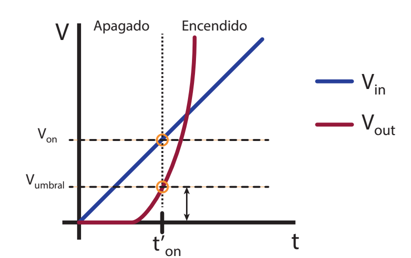


Figure 1. Converter Start-up

  - **Efficiency tests at a constant voltage value and different load values:** In this stage of the sequence, will be possible to analyze the behavior of the converter for different load values.

  - **Converter shut down:** The input voltage of the converter will decrease until reaching shut down voltage level of the converter. In this step of the sequence, the converter shut down voltage level will be registered.

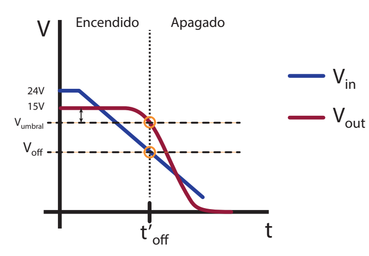


Figure 2. Converter shut down

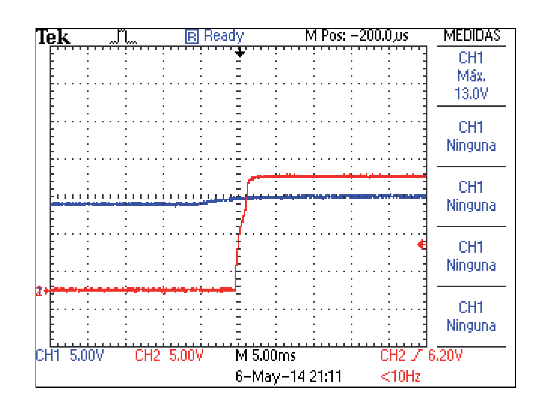
TEST RESULTS:

- **Verification results:**

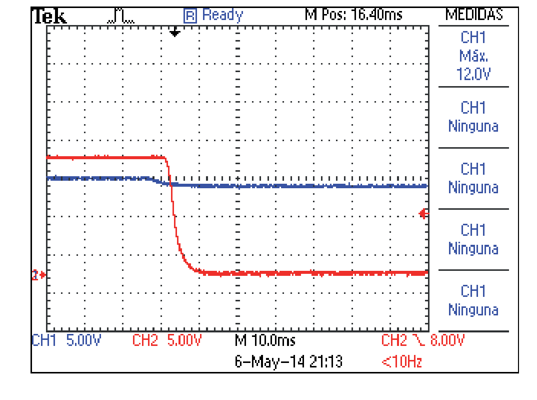
	V_{on} [V]	V_{off} [V]	V_{out} [V]	η_{med}
Espec. Técnicas	12	11	15	84%
Medición	12,17	11,15	15,035	85,25%
Cumple con las especificaciones	✓	✓	✓	✓

Figure 3. Verification results

- **Automated oscilloscope screen capture at the electronic ignition and shut down of the converter.**

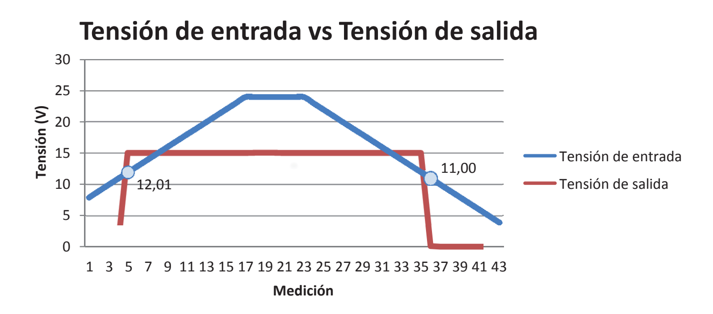


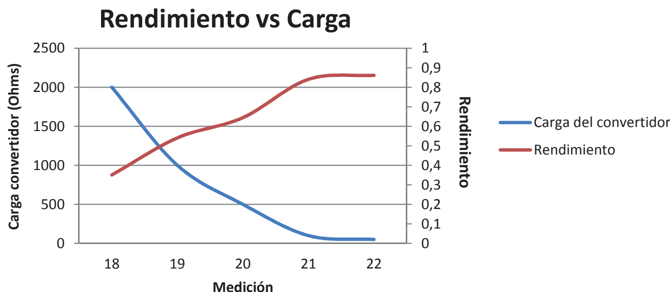
Oscilloscope screen capture at start-up voltage



Oscilloscope screen capture at shut down voltage

- **Analysis of the converter performance for different load values**





DOCUMENTO I

MEMORIA



Índice general

Índice de figuras

Índice de tablas

PARTE I

ESTUDIO DE LA METODOLOGÍA



Capítulo 1

Metodología orientada al desarrollo de sistemas de test

1. Introducción a la metodología. Motivación

La estructura de todo proyecto, queda desde sus inicios, intrínsecamente determinada por el modo en el que éste es planificado, diseñado, construido , etc.

Por metodología se entiende: "Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal." [Rae] Dichos métodos quedan determinados por el tipo de proyecto y por el nivel de normalización (respecto a ciertos estándares u otros proyectos) con el que se llevan a cabo. Un proyecto normalizado o estandarizado es aquel que cumple con las exigencias del cliente siguiendo un modelo previamente definido y que cumple con unos pre-requisitos aprobados por un colectivo a nivel interno de una empresa o incluso a nivel nacional o internacional.

Los objetivos que se persiguen al implementar una metodología orientada al desarrollo de proyectos de manera estándar son los siguientes:

- Una **mayor compatibilidad** entre los distintos elementos que componen un proyecto. Al establecer una metodología normalizada e igual para todos, los componentes pertenecientes a fuentes de trabajo distintas podrán ser compatibles entre sí.

- Poder **evitar problemas ya solventados** ocurridos en casos anteriores. Establecer una metodología de trabajo, teniendo en cuenta los errores ya cometidos, permite reducir el tiempo de pruebas y verificaciones.

- **Aumentar la eficiencia**, tanto en el proceso de diseño como en la ejecución. Conocer de antemano cuál es la estructura básica de la que se debe partir o qué elementos pertenecientes a otros proyectos pueden ser reutilizados, permite agilizar tiempos de diseño y ejecución. Esto solo es posible gracias a la implementación de una metodología compatible entre proyectos y los elementos del mismo.

- **Evitar "puntos ciegos"**. Al mantener una forma de trabajo regida por una metodología robusta, se reduce la probabilidad de ignorar u omitir elementos cruciales en la ejecución del proyecto.

A la hora de aplicar una metodología de trabajo efectiva, pueden llegar a darse diversas situaciones:

- La formación y preparación de cada uno de los operarios no es siempre homogénea, por lo que puede ocurrir que los periodos de adaptación sean mayores o menores dependiendo de sus capacidades y conocimientos.
- La tecnología disponible suele variar dependiendo de cada proyecto. Es altamente recomendable que la metodología a seguir sea flexible a los cambios y pueda adaptarse a cualquier escenario de trabajo.
- Todos los operarios puede que no estén de acuerdo con la implementación de una nueva metodología de trabajo. Transmitir las ventajas y razonar el porqué de dicha implementación a todos los trabajadores, representa una etapa clave dentro del proceso de adaptación.

Con el avance de los medios tecnológicos, la magnitud de los proyectos de ingeniería va en aumento. La única forma viable de poder llevar a cabo tales proyectos, es mediante el reparto de tareas entre los distintos miembros de un equipo técnico.

Es frecuente que cada uno de estos miembros presente hábitos distintos de trabajo: elaborar código, enfocar diferentes soluciones ante un mismo problema, etc., son formas de trabajo que aportan al proyecto matices característicos de cada integrante. El llevar a cabo las distintas partes del proyecto, repartiendo la carga de trabajo entre todos los miembros, presenta numerosas ventajas respecto a la eficiencia y rendimiento de un proyecto individual.

Una de las mayores dificultades puede aparecer durante la puesta en común de cada componente del proyecto. Si durante el diseño no se ha aplicado una normalización común para todas, es muy probable que lleguen a producirse serios problemas de compatibilidad entre los distintos componentes. La resolución de dichas incompatibilidades puede inducir a serios retrasos en la fecha de entrega o incluso a un mal funcionamiento del sistema en cuestión. Con el fin de evitar cualquier situación de incompatibilidad, se emplearán metodologías de trabajo que resuelvan a priori cualquier ambigüedad o posible incongruencia entre los distintos elementos de enlace de cada una de las partes del proyecto.

El diseño de cada metodología de trabajo no siempre es el mismo debido a que cada sección dentro de una empresa o incluso cada proyecto puede llegar a requerir un modo de trabajo diferente. Recordemos que la metodología debe ser la pieza angular que permita que elementos independientes sean totalmente compatibles entre sí.

Una vez iniciado el proyecto y establecida una metodología concreta, ésta será la que vaya configurando la planificación, diseño, construcción, etc. Y no al revés. Por el contrario, dichas etapas dentro del proyecto serán las que perfeccionen la metodología existente, haciéndola más eficiente y flexible a cada situación.

2. Características generales y fases de desarrollo de una metodología orientada al diseño de sistemas de test

Para poder comenzar a diseñar la metodología que posteriormente será implementada en el diseño de sistemas de test, se deberán determinar aquellos aspectos que estén relacionados con [1]:

- Métricas empleadas en el estudio del sistema de test que se quiere diseñar.
- Herramientas utilizadas por y para el diseño del sistema de test.
- Actividades o pruebas que se llevarán a cabo en cada secuencia de test.
- Rol que desempeñará cada ingeniero en la ejecución del proyecto.

Para determinar dichos aspectos de manera eficiente, se requiere una temprana gestión de medios y recursos. Por recursos podemos entender: "Aquellos servicios, sistemas de hardware, software, bases de datos, experiencias previas en proyectos anteriores, etc., que puedan ser de utilidad en el diseño y la ejecución de dicho proyecto".

Una correcta gestión favorece, en todos sus aspectos, la ejecución del proyecto. Poder anticiparse y planificar los recursos y medios requeridos permite, una mejor estimación del coste total del proyecto y del tiempo medio de ejecución del mismo.

De entre todos los posibles roles previstos para el diseño de un sistema de test se puede destacar, basándose en los niveles de criticidad presentes en su función:

- **Responsable de testing:** Encargado de la depuración y verificación del software destinado al comandado del sistema de test. Una verificación incorrecta o poco exhaustiva puede ocasionar el mal funcionamiento del sistema.
- **Responsable de gestión y planificación de pruebas:** Encargado del diseño de la arquitectura de cada una de las pruebas programadas para el sistema de test, ya en un estado funcional. Una incorrecta programación del sistema de test puede ocasionar resultados inverosímiles, dando por bueno un producto defectuoso, o derivar en pérdidas económicas al verse obligados a desechar componentes en buenas condiciones.

2.1. Fases propias del diseño y planificación de una metodología

La fases propias del diseño y planificación de una metodología serán [1]:

2.1.1. Fase inicial o de identificación

Los **objetivos** que se pretenden alcanzar en esta primera etapa serán los siguientes:

- Identificar las **herramientas y participantes** que formarán parte del equipo de trabajo.
- Identificar **riesgos potenciales**.
- Identificar **experiencias previas** con proyectos anteriores.

2.1.2. Fase de formación:

Las **tareas** principales de esta etapa serán:

- Capacitar a cada uno de los miembros del equipo técnico que llevarán a cabo la ejecución del proyecto.
- Establecer un modelo de trabajo cooperativo entre todos los implicados en los distintos proyectos piloto o proyectos en pruebas, con el fin de poder identificar las prácticas que se llevarán a cabo, definir y establecer la ejecución de cada parte del proyecto, detectar posibles problemas y asegurar una correcta utilización de las herramientas de test.
- Establecer periódicamente reuniones, donde cada miembro del grupo pueda exponer nuevas ideas, sugerencias y conocimientos adquiridos.

El verdadero sentido de esta etapa quedará definido en los siguientes **objetivos**:

- Asignar las distintas tareas y funciones de cada participante dentro del proyecto.
- Identificar cada una de las partes del proyecto.
- Consolidar las herramientas de hardware y software que se deberán utilizar en el diseño del sistema de test.
- Capacitar al personal mediante una formación continua para que pueda llevar a cabo la ejecución de la tarea asignada.
- Definir las métricas que serán empleadas por todos y cada uno de los miembros del grupo, a fin de consolidar un modelo de proyecto estándar para todos.

2.1.3. Fase de consolidación:

Una vez finalizada la fase de planificación del proyecto, es necesario **consolidar una única metodología**. Para ello se deberá:

- Valorar y aplicar las posibles mejoras planteadas en las distintas reuniones y juntas.
- Confirmar las métricas que serán empleadas.
- Establecer un modelo de control sobre los tiempos de ejecución y entrega del proyecto.

Los **objetivos** que se persiguen en esta fase serán los siguientes:

- Conseguir una mayor autonomía en cada uno de los participantes del grupo, autogestionando la parte del proyecto que les ha sido asignada.
- Resolver posibles problemas de compatibilidad entre las distintas partes del proyecto.

2.1.4. Fase final o de implantación:

Una vez llegado a un acuerdo y consolidada ya la metodología, se procede a su implantación en todos aquellos proyectos cuya arquitectura se asemeje a aquella para la cual fue inicialmente diseñada.

2.2. Objetivo Global

El principal objetivo que se persigue al implantar una nueva metodología es el siguiente: *"Conseguir una elevada calidad en la ejecución del proyecto, disminuyendo costes y aumentado la eficiencia"*.

3. Características específicas de metodologías aplicadas en el diseño de software de sistemas

Actualmente existen numerosas metodologías aplicadas en el desarrollo de arquitecturas de software. En este proyecto se emplearán conjuntamente tres tipos de metodologías distintas.

Entre todas las metodologías existentes cabe destacar, de forma esquemática, las tres más representativas y utilizadas en este proyecto [2]:

- **Metodologías estructuradas:**
  - Orientada a procesos.
  - Orientada a datos.
    - Estructura jerárquica.
    - Estructura no jerárquica.
  - Estructura mixta.
- **Metodología orientada a objetos [OO].**
- **Metodología implementada en sistemas en tiempo real.**

Como se explicará en los siguientes apartados, cada una de estas metodologías será aplicada en aquellas partes de la arquitectura del software donde mejor satisfagan las especificaciones técnicas, tanto en la gestión de procesos como en la eficiencia con la que se lleva a cabo el procesado de datos. Las razones por las que se llegó a la decisión de utilizar dichas metodologías, quedan notoriamente justificadas al presentar las principales características de cada uno:

3.1. Metodologías Estructuradas

Están basadas en una estructura "Top - Down". Define las especificaciones del sistema de test mediante una descomposición funcional del mismo.

Dicha descomposición, permite una visualización estructurada del conjunto de especificaciones mediante la utilización de gráficos particionados, descendentes y jerárquicos de los distintos procesos del sistema.

Este tipo de metodologías serán empleadas en el diseño de secuencias y campañas (conjunto de secuencias) que se llevarán a cabo una vez se encuentre el sistema de test en modo funcional.

Está compuesta por:

- **Diagrama de flujo de datos**. Cada diagrama representa los distintos procesos que se llevan a cabo y del mismo modo, se representa también el flujo de información existente entre las distintas funciones.
- **Diccionario de datos:** Contienen las distintas referencias sobre el flujo de información reflejado en el diagrama.
- **Especificaciones de procesos:** Análisis en profundidad de cada función y proceso que componen el proyecto.

3.2. Metodologías Orientadas a Objetos [OO]

En esta metodología, tanto los procesos como las variables serán tratados como un único componente u objeto. Cada sistema puede estar compuesto por varios objetos capaces de interactuar entre sí.

Los fundamentos de una metodología OO son:

- **Abstracción:** Descripción "superficial" de los elementos, funciones o métodos de un objeto (*Clase abstracta de un objeto*).
- **Encapsulación:** Se agrupan los elementos pertenecientes a los objetos, de forma que el usuario solo tenga acceso a aquellos elementos que puedan ser configurables, mientras que el resto permanecerán ocultos para el usuario. (*Métodos contenidos dentro de funciones de propósito general*).
- **Modularidad:** Propiedad que permite a cada objeto el poder estar dividido en distintos métodos, procesos, funciones, etc.
- **Jerarquía o herencia:** Estructura lógica presente en los distintos niveles de abstracción.
- **Tipificación:** Cada objeto estará propiamente definido para evitar cualquier posible suplantación entre objetos distintos.
- **Persistencia:** Propiedad presente en aquellos objetos que permanecen activos después de que el espacio de nombres donde fue instanciado haya cambiado o incluso desaparecido (*Declaración de variables en niveles superiores*).

3.3. Metodologías Orientadas a Sistemas en Tiempo Real

Presente en sistemas cuya prioridad es el control global de todos los procesos. Esta metodología estará comúnmente implantada en sistemas donde sea necesario:

- Gestionar la concurrencia de objetos de diversa naturaleza.
- Priorizar la ejecución de unos procesos sobre otros.
- Establecer flujos de información entre tareas de forma síncrona o asíncrona.

- Manejar interrupciones durante la ejecución de procesos simultáneos.
- Establecer un marco de referencia temporal para cada uno de los procesos que han de ejecutarse.

4. Metodologías de Verificación y Pruebas de Test

Para poder establecer una correcta arquitectura de software, es necesario conocer cuál será el tipo de prueba que será llevada a cabo en el sistema de test.

Conocer las distintas metodologías aplicadas a procesos de verificación, permite poder valorar cuál de ellas se adapta mejor a las especificaciones técnicas que han de verificarse en cada prueba.

Entre todas las posibles pruebas de verificación/validación, se pueden destacar algunas de las más comúnmente utilizadas [3]:

- **Pruebas automatizadas:** Cada una de dichas pruebas, como su nombre indica, se gestiona de forma automática sin la necesidad de una participación directa por parte del operario.
- **Pruebas de aceptación:** Comprueban si un producto cumple con las especificaciones técnicas del cliente. Es frecuente que dichas pruebas sean supervisadas por el cliente a la hora de aceptar o no la valoración de los resultados.
- **Pruebas de "Caja Negra":** Son comúnmente aplicadas en dispositivos donde el funcionamiento interno es desconocido. La valoración de los resultados se basa en analizar las salidas generadas al estimular el dispositivo con unas entradas conocidas previamente definidas.
- **Pruebas de "Caja Blanca":** El fundamento de estas pruebas se basa en el análisis del funcionamiento, procesos y estructuras internas que configuran el cuerpo de la aplicación o dispositivo.
- **Pruebas de compatibilidad:** Miden el nivel de operatividad conjunta entre dos o más dispositivos, hardware-software, etc. Este tipo de pruebas pueden ser llevadas a cabo de forma automática o manual.
- **Pruebas de normalización:** Su objetivo será el de determinar si se cumple o no con los actuales requisitos y estándares presentes en un sector determinado.
- **Pruebas de exploración:**También conocidas como pruebas "on the fly". Son pruebas realizadas sobre la marcha, que se centran en una función específica de la aplicación, un componente concreto del dispositivo, etc. Las pruebas de exploración son llevadas a cabo por aquellos operarios altamente cualificados que conocen el posible riesgo de existencia de errores en puntos concretos del sistema. Puede ocurrir que la finalidad de la prueba no sea comprobar el número de errores, sino verificar que la fiabilidad del sistema asegura una tasa de errores por debajo de los niveles críticos.
- **Pruebas de funcionalidad:** Se revisa que el dispositivo cumple con todas las especificaciones y que todas sus funciones operan correctamente. Las pruebas llevadas a cabo comprueban el correcto funcionamiento componente a componente, verificando su comportamiento tanto en circunstancias normales como anómalas.

- **Pruebas regresivas:** Orientadas a la depuración de errores o al mantenimiento de dispositivos. Su estructura es similar a las pruebas funcionales. Su objetivo se centra en comprobar que, una vez se realizan los cambios de depuración, tanto el elemento depurado como el resto de elementos funcionan correctamente.
- **Pruebas de estrés o aguante:** Se basan principalmente en someter al dispositivo a condiciones extremas de trabajo y analizar qué niveles de estrés es capaz de soportar, determinando si dichos niveles están por encima o por debajo del umbral establecido en las especificaciones técnicas. Estas pruebas son muy frecuentes en sistemas de vuelo.

Capítulo 2

Metodología aplicada a un caso real

1. Objetivo:

Diseñar un sistema de test totalmente automatizado, que permita verificar desde tarjetas de potencia hasta componentes electrónicos.

Para llevar a cabo el proyecto, se implantarán las metodologías previamente descritas, aplicadas tanto en la ejecución del mismo como en el diseño de cada función y de cada secuencia que componen en su conjunto el sistema de test.

2. Métricas establecidas

Para poder empezar con la ejecución del proyecto, es necesario establecer las métricas necesarias para determinar el plazo de entrega del proyecto, las funciones del sistema de test, la arquitectura de ficheros, etc.

3. Planificación del proyecto

Para poder cumplir los plazos de entrega, es necesario establecer un riguroso control sobre el tiempo destinado a cada una de las etapas que conforman el proyecto:

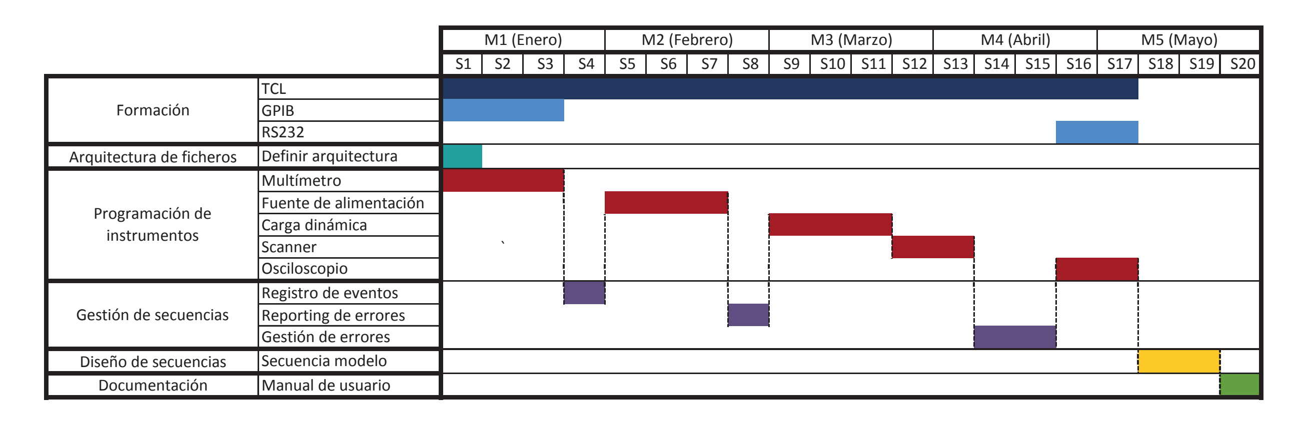


Figura 1. Cronograma establecido para el desarrollo del sistema de test

4. Especificaciones de software

4.1. Un único lenguaje de programación

Con el objetivo de unificar el lenguaje de programación empleado, tanto en el comandado de instrumentos como en el diseño de las secuencias de test, se decantó por utilizar un lenguaje que tuviera, entre otras muchas propiedades, un soporte de librerías compatible con la comunicación entre el ordenador y cada uno de los instrumentos del sistema de test (Comunicación GPIB, RS232, USB, ect.), que presentara una gran escalabilidad en cuanto a dispositivos y secuencias programadas, fuera un lenguaje robusto y fácil de usar, etc.

4.2. Reusabilidad de estructuras

Cada librería destinada al comandado de un instrumento debe presentar una estructura estándar para todas y cada una ellas. De esta forma, se posibilita el poder reutilizar gran parte del código, en el caso de que se sea necesario comandar un instrumento con la misma función pero de distinta marca o modelo al ya programado.

4.3. Estructura de ficheros y datos

El objetivo principal que se persigue al establecer una estructura fija en la jerarquía de ficheros es la creación de un software destinado al comandado del sistema de test, que sea completamente autónomo y permita su potabilidad entre distintos sistemas.

Estableciendo una arquitectura fija, se está facilitando la implementación de nuevas librerías de comandado al sistema de test. Simplemente combinando las carpetas que ya contiene el sistema de test actual con los ficheros que contienen los nuevos módulos, se consigue de forma fácil y sencilla un nuevo sistema ya ampliado, totalmente operativo, sin necesidad de realizar ningún cambio en las direcciones de trabajo.

La estructura de ficheros empleada será la siguiente:

4.3.0.1. System \ documents

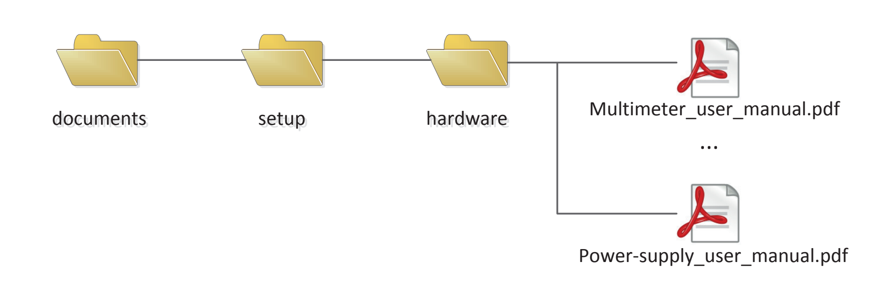


Figura 2. Arquitectura de ficheros: “System \ documents”

La carpeta “documents” contendrá toda la documentación relevante acerca de los instrumentos que componen el sistema de test: *datasheet, user-manual,* etc.

En el caso de que el usuario quiera conocer cual es la función y la correcta utilización de cada dispositivo, deberá buscar en dicha carpeta.

4.3.0.2. System \ setup

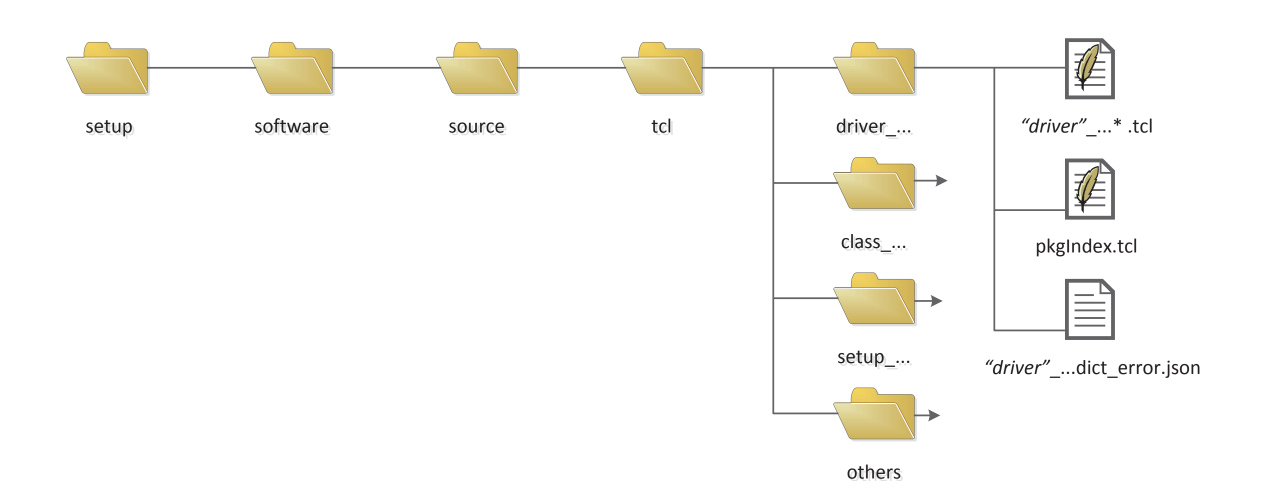


Figura 3. Arquitectura de ficheros: “System \ setup”

La carpeta “setup” contiene todas las librerías que componen el sistema de test. Cada una de las librerías se nombrará de la siguiente forma dependiendo de su función:

- **driver_***: Contiene los driver de comandado del instrumento (*).
- **class_***: Contiene la clase abstracta correspondiente al instrumento (*)
- **setup_***: Contiene el “setup” o inicialización de la prueba de test (*)
- ***Otras librerías***: Contienen las librerías de distinta naturaleza a las anteriores. Estas carpetas no contendrán ningún tipo de prefijo adicional.

4.3.0.3. System \test

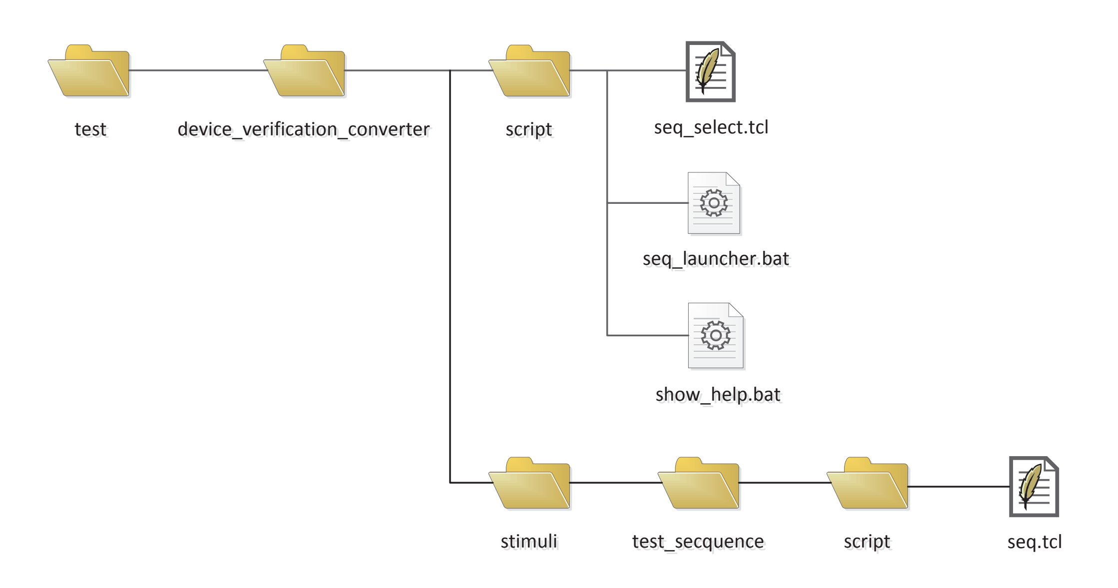


Figura 4. Arquitectura de ficheros: “System \ test”

El directorio “test” contiene dos grupos de ficheros dentro de la carpeta correspondiente a la prueba de verificación “device_verifiation_converter”:

- **\ script**: Contiene los ficheros correspondientes a la selección del modo de ejecución de la prueba. “Modo secuencia” o “Modo depuración”
  - Modo secuencia: Comandado automático del instrumento a partir de una secuencia previamente establecida.
  - Modo depuración: Comandado manual del instrumento mediante comandos.
- **\ stimuli**: Contiene la secuencia de test correspondiente a la prueba.

4.3.0.4. System \ tools



Figura 5. Arquitectura de ficheros: “System \ tools”

El fichero “userwork.bat” carga la configuración en el sistema de todos los directorios de trabajo: Directorio de aplicaciones (Tcl, Doxigen...), librerías por defecto, workpath, etc.

4.3.0.5. System \ work

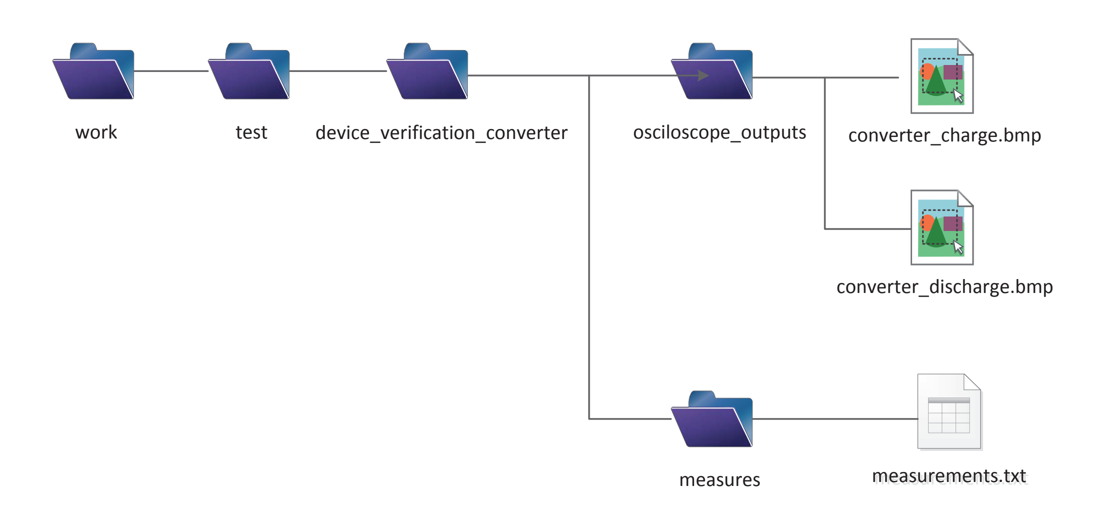


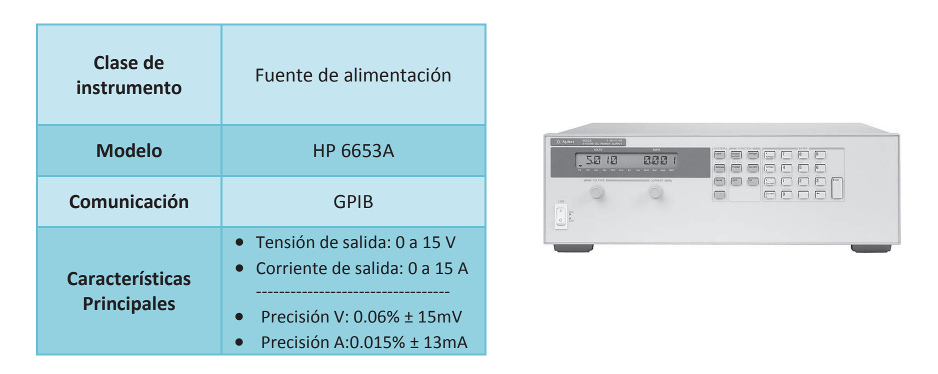
Figura 6. Arquitectura de ficheros: “System \ work”

La carpeta “work”, a diferencia del resto de ficheros creados por el usuario de forma manual, estará generada automáticamente tras la ejecución de cada secuencia de test. En ella estarán contenidos todos los informes, capturas del osciloscopio, medidas realizadas, etc.

5. Herramientas que componen el sistema de test

Los instrumentos que componen actualmente el sistema de test son los siguientes:

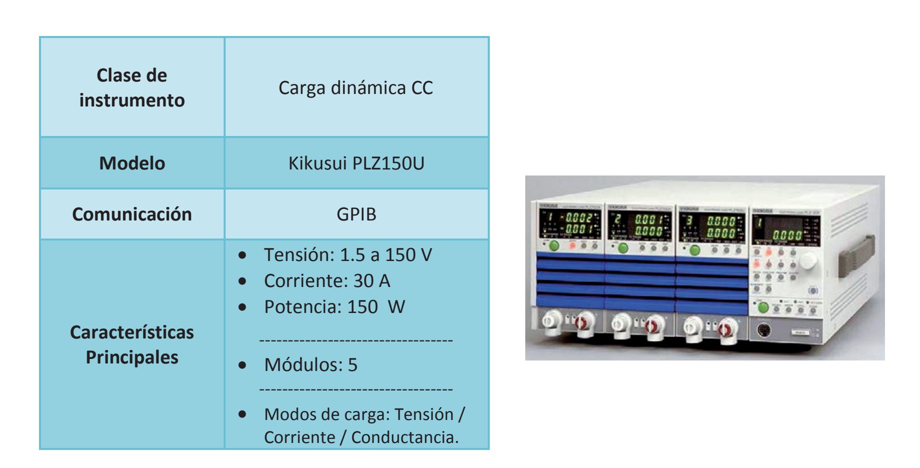
5.1. HP 6653A [Fuente de alimentación]



Clase de instrumento	Fuente de alimentación
Modelo	HP 6653A
Comunicación	GPIB
Características Principales	• Tensión de salida: 0 a 15 V • Corriente de salida: 0 a 15 A ---------------------------------- • Precisión V: 0.06% ± 15mV • Precisión A:0.015% ± 13mA

Figura 7. HP 6653A [Fuente de alimentación

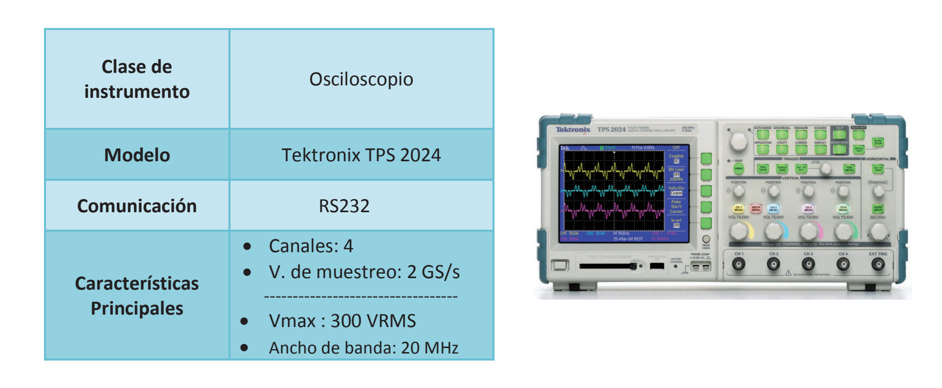
5.2. Kikusui PLZ150U [Carga dinámica]



Clase de instrumento	Carga dinámica CC
Modelo	Kikusui PLZ150U
Comunicación	GPIB
Características Principales	• Tensión: 1.5 a 150 V • Corriente: 30 A • Potencia: 150 W ---------------------------------- • Módulos: 5 ---------------------------------- • Modos de carga: Tensión / Corriente / Conductancia.

Figura 8. Kikusui PLZ150U [Carga dinámica]

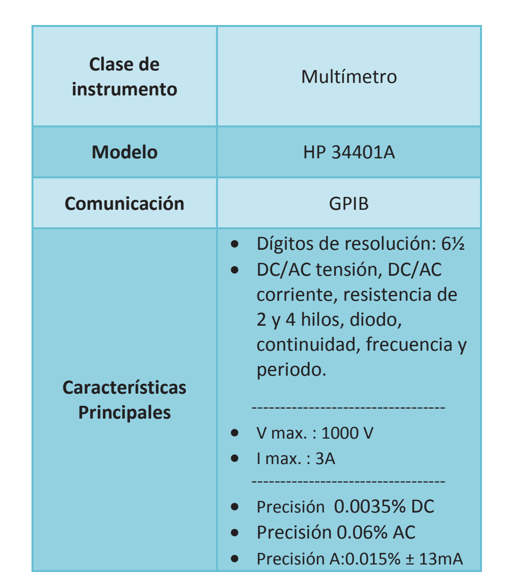
5.3. Tektronix TPS2024 [Osciloscópio]



Clase de instrumento	Osciloscopio
Modelo	Tektronix TPS 2024
Comunicación	RS232
Características Principales	• Canales: 4 • V. de muestreo: 2 GS/s ---------------------------------- • Vmax : 300 VRMS • Ancho de banda: 20 MHz

Figura 9. Tektronix TPS2024 [Osciloscópio

5.4. HP 34401A [Multímetro]



Clase de instrumento	Multímetro
Modelo	HP 34401A
Comunicación	GPIB
Características Principales	• Dígitos de resolución: 6½ • DC/AC tensión, DC/AC corriente, resistencia de 2 y 4 hilos, diodo, continuidad, frecuencia y periodo. ---------------------------------- • V max. : 1000 V • I max. : 3A ---------------------------------- • Precisión 0.0035% DC • Precisión 0.06% AC • Precisión A:0.015% ± 13mA

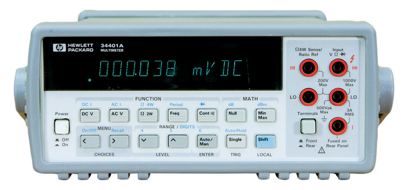
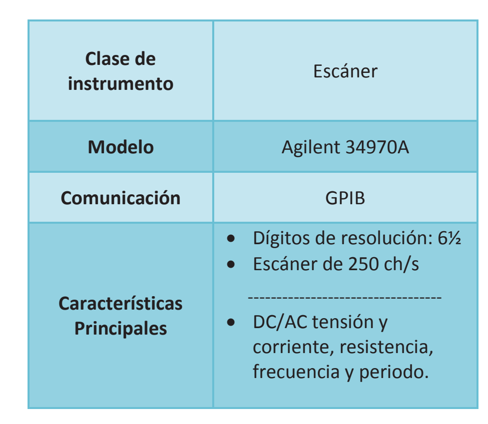


Figura 10. HP 34401A [Multímetro

5.5. Agilent 34970A [Escáner]



Clase de instrumento	Escáner
Modelo	Agilent 34970A
Comunicación	GPIB
Características Principales	• Dígitos de resolución: 6½ • Escáner de 250 ch/s ---------------------------------- • DC/AC tensión y corriente, resistencia, frecuencia y periodo.



Figura 11. Agilent 34970A [Escáner]

*Las imágenes mostradas pueden tener derechos de copyright de sus respectivas marcas.
*HP, Agilent, Kikusui y Tektronix son marcas registradas.
*Las imágenes pueden no corresponder con el instrumento utilizado en el sistema de test.
*Si desea información mas detallada sobre los instrumentos utilizados, se adjunta enlace en la *bibliografía de herramientas e instrumentos utilizados*.

6. Especificaciones de la secuencia de test

Para comprobar el correcto funcionamiento del sistema de test, se llevará a cabo la verificación de un componente electrónico basándose en las especificaciones técnicas indicadas en el datasheet del fabricante.

6.1. Dispositivo sometido a la prueba de test

El dispositivo bajo test será un convertidor CC/CC que presenta las siguientes especificaciones técnicas indicadas en su datasheet correspondiente:

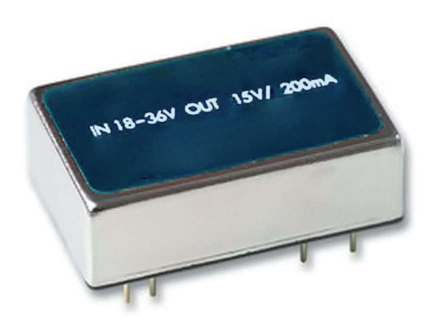


Figura 12. Convertidor CC/CC

Tensión de entrada nominal	24 VDC
Rango de tensión de entrada	18-36 VDC
Tensión de salida	15 VDC
Corriente de salida máxima	200 mA
Eficiencia media	84 %

Corriente de entrada sin carga / con carga	5 mA / 150 mA
Tensión mínima de encendido **Tensión límite de apagado**	12 VDC 11 VDC
Precisión de voltaje de salida	±1 %

Tabla 1. Especificaciones técnicas del convertidor

6.2. Diseño de la prueba

En el diseño de cualquier prueba de test se debe considerar cualquier situación de riesgo: primero para el operario y segundo para el dispositivo o sistema de test. Para evitar dichas situaciones se deberán emplear sistemas de protección, detección de fallos en la comunicación con los instrumentos, etc.

Para poder establecer cada una de las etapas que componen la verificación del dispositivo, es recomendable conocer de forma aproximada el comportamiento del dispositivo en cada una de las fases de la prueba.

6.2.1. Estudio del arranque del convertidor

Se sabe a ciencia cierta que todo convertidor necesita ser alimentado a una tensión mínima de encendido para poder empezar a trabajar como tal.

La primera parte de la secuencia de verificación se centrará en el análisis de esta primera etapa del convertidor. Se irá incrementando la tensión de entrada como una rampa ascendente

hasta que se detecte el encendido del mismo. Para detectar que realmente el convertidor se ha encendido, se establecerá una tensión umbral por encima de la cual se considerará que ya está en funcionamiento. Se aproximará el tiempo de encendido por el momento en el que la tensión de salida supere la tensión umbral (t'_{on}).

Se estima que la gráfica correspondiente al momento de encendido del convertidor tenga aproximadamente la siguiente forma:

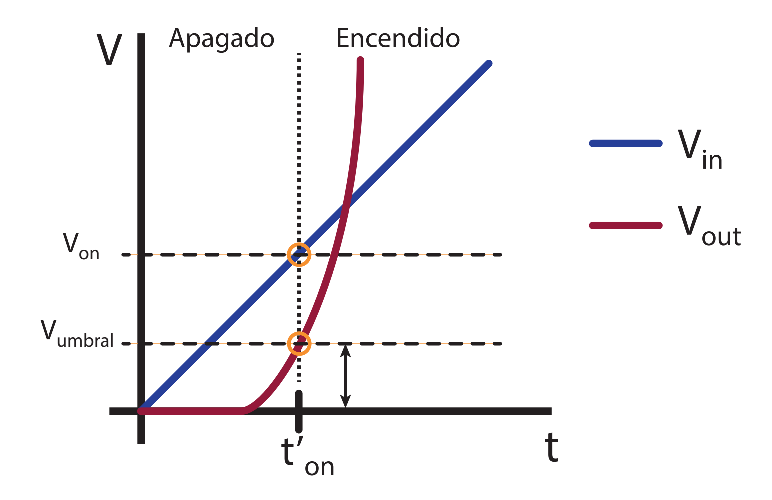


Figura 13. Arranque de un convertidor

6.2.2. Pruebas de rendimiento a tensión nominal constante y carga variable

Una vez que el convertidor se encuentre trabajando en régimen permanente, con una tensión de salida de 15 V y esté alimentado a tensión nominal (24 V), se realizarán las distintas pruebas de rendimiento correspondientes a distintos valores de carga.

Para el cálculo del rendimiento se empleará la siguiente fórmula:

$$\eta = \frac{P_{out}}{P_{out} + P_{in}} = \frac{(V_{out} \cdot I_{out})}{(V_{out} \cdot I_{out}) + (V_{in} \cdot I_{out})}$$

6.2.3. Estudio del apagado del convertidor

Una vez completada la secuencia correspondiente a las pruebas de rendimiento, se iniciará la tercera y última etapa de la verificación.

Partiendo de la tensión de alimentación nominal, se irá reduciendo el voltaje a la entrada del convertidor hasta asegurarnos de que se ha producido la desconexión del mismo. Momento en el que registraremos la tensión a la que se ha apagado el convertidor.

Se estima que la gráfica correspondiente al momento de apagado del convertidor tenga aproximadamente la siguiente forma:

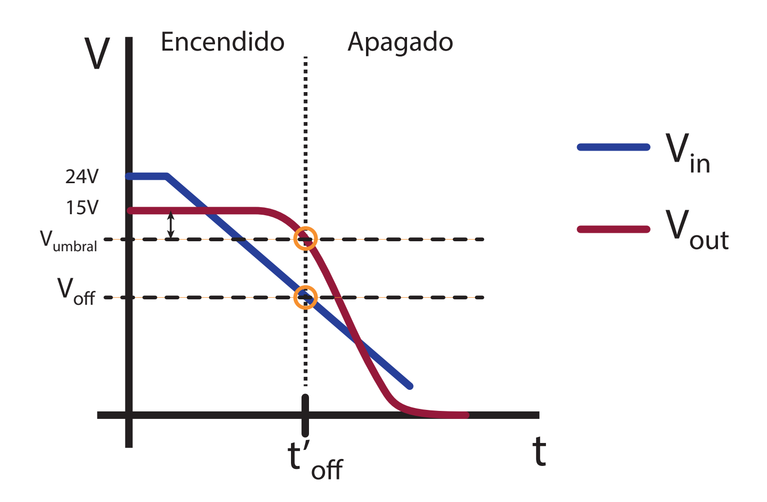


Figura 14. Desconexión de un convertidor

6.3. Procesado de medidas. Resultados

Se diseñará una plantilla de mediciones que permita agrupar los resultados obtenidos de forma estructurada y bien organizada:

	Configuración de fase		Mediciones						
Fase	Tensión	Carga	V_{in}	I_{in}	V_{out}	I_{out}	P_{in}	P_{out}	η
1	Rampa ascendente	Constante							
2	Constante	Incrementada							
	Constante	Reducida							
4	Rampa descendente	Constante							

Tensión de histéresis de carga:
Tensión de histéresis de descarga:

Resultado de la prueba:

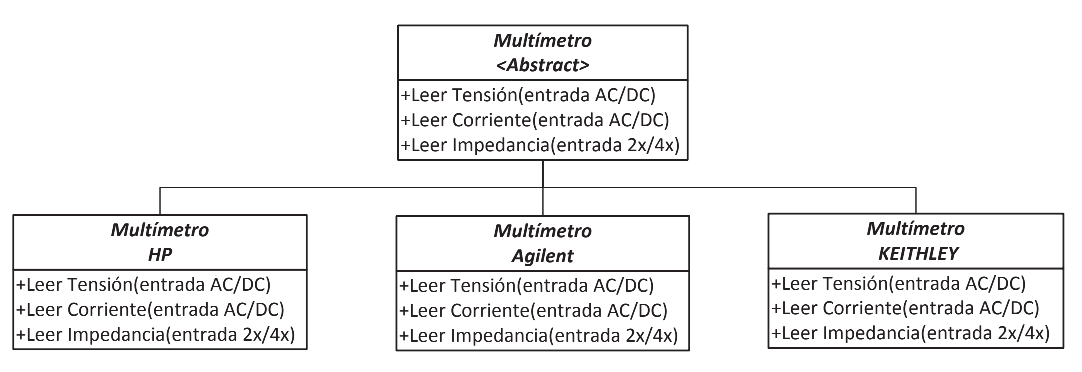
Tabla 2. Plantilla de resultados

Una vez completada la tabla con las mediciones, se comparan los resultados obtenidos con los esperados. Si los resultados se encuentran dentro de los umbrales admisibles, puede afirmarse que el convertidor se mantiene dentro de sus condiciones nominales de funcionamiento.

7. Arquitectura del software

7.1. Librerías de comandado de instrumentos

Para gestionar de manera eficiente todos los instrumentos y las funciones de cada uno de ellos, es necesario implementar una metodología "OO" (Orientada a objetos) que permita descomponer cada tipo de instrumento en clases abstractas de las que hereden las clases asignadas a cada modelo.



* HP, KEITHLEY y Agilent son marcas registradas.

Figura 15. Ejemplo de clase abstracta

Utilizar clases abstractas permite unificar todas las funciones pertenecientes a un mismo tipo de instrumento. De esta forma, el usuario solamente necesitará conocer cuales son los comandos de la clase abstracta, pues serán los mismos para el resto de instrumentos.

7.2. Registro de comandos y “reporting” de errores

Con el objetivo de poder gestionar el flujo de información que procesa el sistema, se requiere de funciones capaces de registrar cada uno de los comandos que se envían por cada puerto, incluyendo la hora en que fue enviado. De manera similar, se precisan librerías dedicadas al “reporting” de errores que permitan al usuario determinar qué errores se han producido y cuales de ellos son críticos a la ahora de gestionar la respuesta del sistema ante dicho error.

Tanto el registro de comandos como el “reporting” de errores serán diseñados siguiendo una metodología “OO”. Las funciones que componen ambas estructuras serán llamadas directamente desde las funciones de comandado de cada instrumento.

7.3. “Error Handle” o Gestor de errores

Gestionar la actuación del sistema en situaciones en las que se ha producido algún error durante la secuencia forma parte de las funciones pertenecientes a lo que comúnmente se conoce como “Error Handle” o “Gestor de errores”. Ambas funciones serán instanciadas siguiendo una metodología “OO” como “objetos” independientes, controlados a bajo nivel en cada función de cada instrumento.

La clase “Error Handle” se encarga de, dependiendo de la severidad del error notificado por la función “Reporting de Errores”, ejecutar el protocolo de seguridad programado. Por defecto, el protocolo de actuación ante errores que presenten una severidad crítica se basa en una parada controlada de todos los dispositivos. Dicha actuación podrá ser reprogramada por el operario en función de la prueba de test que se esté llevando a cabo.

Los principales objetivos del gestor de errores, como ya se indicó previamente, serán: proteger la salud de los operarios como prioridad número uno y posteriormente, proteger el equipo de test y el dispositivo sometido a pruebas de cualquier situación que pueda producir daños irreversibles en los mismos.

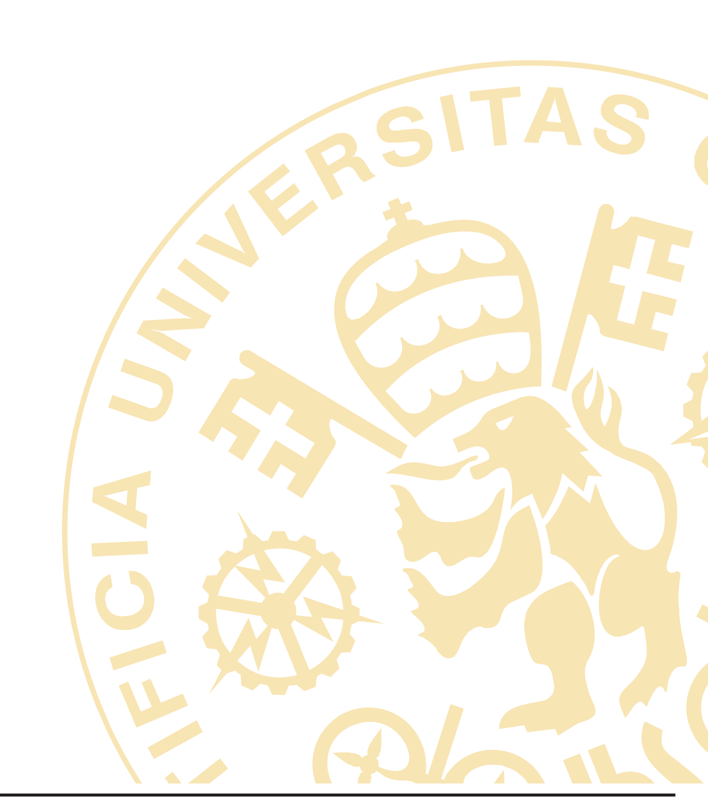
7.4. Secuencias de test

Para poder diseñar una secuencia de test, se requiere configurar al menos dos elementos: La SECUENCIA, que contiene el comandado de instrumentos y el procesamiento de los datos procedentes de las medidas, y el SETUP, que contiene la inicialización de los instrumentos. Tanto la secuencia como el setup serán programados siguiendo una metodología estructurada, es decir, su estructura presentará una ejecución secuencial del tipo “Top - Down”.

En pruebas de test avanzadas, puede darse la necesidad de tener que ejecutar varios procesos al mismo tiempo. Para ello se emplearán métodos de ejecución en paralelo, tales como el uso de hilos, interrupciones, etc. que requieran de una metodología orientada a sistemas en tiempo real.

PARTE II

SISTEMAS DE TEST



Capítulo 1

Pasado, presente y futuro de los sistemas de test

1. Introducción a los sistemas de test. Motivación

Los procesos de verificación y sistemas de test no siempre han presentado unos niveles de automatización tan elevados como los actualmente presentes en fábricas de producción en serie o empresas dedicadas al diseño de componentes o dispositivos electrónicos. Dichos procesos de verificación pueden basarse en la comprobación de unas especificaciones técnicas específicas o simplemente en la calidad de un producto.

En electrónica, una secuencia de test se basa esencialmente en la medición de las respuestas que genera el dispositivo bajo test, al aplicarle unas entradas o estímulos determinados. Una vez realizadas las mediciones, se procesa dicha información, extrayendo las conclusiones que determinen si el dispositivo ha pasado o no la prueba.

La automatización de las secuencias va adquiriendo una mayor relevancia en aquellos entornos de trabajo donde se busca conseguir un alto nivel de reproducibilidad con cada prueba y un tiempo de ejecución lo más breve posible.

Existen entornos de trabajo donde unos altos niveles de automatización pueden llegar a ser innecesarios, debido a una elevada inversión inicial que haga encarecer el coste de cada prueba sin reflejar un ahorro sustancial a largo plazo.

Conocer qué partes de la prueba de test necesitan ser cubiertas mediante procesos automáticos y la compatibilidad del equipo del que se dispone con dicha automatización puede ser un buen comienzo para decidir qué nivel de automatización se adapta mejor a nuestras necesidades.

2. Sistema de test manual

Este tipo de sistemas de test será el primero en ser implementado a nivel industrial. La escasa complejidad propia de los primeros sistemas electrónicos permitía a este tipo de sistemas obtener resultados óptimos de manera eficiente.

En la actualidad, estos entornos de test se emplean en aquellos proyectos caracterizados por [6]:

- Requerir una "velocidad" de secuencia relativamente baja. El tiempo de muestreo de cada medición no es un factor crítico.
- Tener una configuración modulable o abierta, es decir, que pueda estar sometido a modificaciones con relativa frecuencia (Baja reproducibilidad).
- Las mediciones que se llevarán a cabo en dicho proyecto son, en su mayoría, verificaciones inmediatas (Ej: tensión, corriente, etc., en puntos concretos en momentos puntuales).

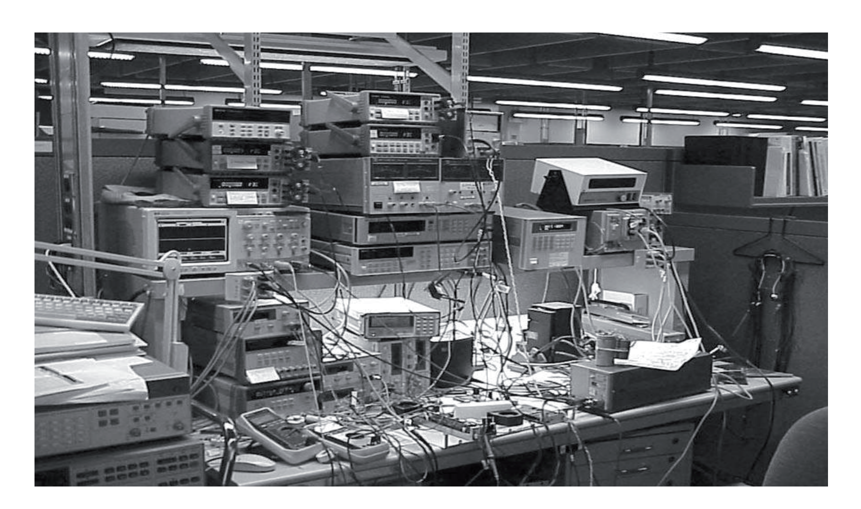


Figura 16. Entorno de trabajo de un sistema de test manual[6]

Para la debida y correcta utilización de los dispositivos en cada una de las pruebas, se requiere de operarios con una elevada destreza en el manejo de dichos instrumentos. La recogida y análisis de los resultados obtenidos en cada prueba puede llegar a ser una tarea larga y, en algunos casos, complicada.

Este tipo de entornos de test es comúnmente utilizado en pequeñas y medianas empresas cuyos trabajos están orientados a la reparación de sistemas o a la verificación de prototipos a pequeña y mediana escala. El coste total medio es, sin duda, el más económico de todos los posibles sistemas de test.

Las secuencias de test ejecutadas de forma manual presentan una gran ineficiencia en proyectos en los que configurar todos los equipos de medición para cada prueba representa un elevado porcentaje del tiempo de éstas.

3. Sistema de Test Semi-Automático

Desde hace más de una década, las empresas pertenecientes al sector eléctrico y de la electrónica se han visto en la necesidad de implementar nuevos entornos de test que les permitan poder realizar pruebas más exhaustivas de forma más rápida y eficiente.

Esta adaptación llevaría a los laboratorios de verificación de sistemas a desarrollar e implantar una nueva forma de trabajo, basada en un entorno de test cuya característica principal será el

comandado de los instrumentos desde el sistema de control y, en ciertas fases de la prueba, también de forma manual.

Los proyectos donde es frecuente encontrar sistemas de test semi-automáticos son:

- Aquellos que, al automatizar las pruebas de test, generan un beneficio por encima de los costes derivados de dicha automatización.
- La magnitud del proyecto no requiere de una plena automatización.
- Los resultados obtenidos deben ser reproducibles.
- Requieren de un sistema de test con cierta flexibilidad.



Figura 17. Entorno de trabajo de un sistema de test semi-automático [6]

Al igual que los sistemas de test manuales, éstos también requieren de un operario experimentado, pues la realización de pruebas de manera semi-automática debe ser monitorizada por un experto y seguir unas normas de seguridad.

El coste total de un sistema de test aumenta cuanto mayor sea el nivel de automatización del mismo. Este aumento en el coste se ve compensado con una notable reducción en el tiempo de ejecución de cada prueba. La automatización de estos procesos suele llevarse a cabo mediante programas basados en una interfaz gráfica muy intuitiva, que permite configurar el instrumento remotamente. A su vez, es bastante frecuente que dicha automatización quede limitada a una ventana de comandos, desde donde se pueda controlar el instrumento mediante el envío de líneas de código una a una, sin posibilidad de poder establecer una secuencia lógica que permita la interacción entre instrumentos.

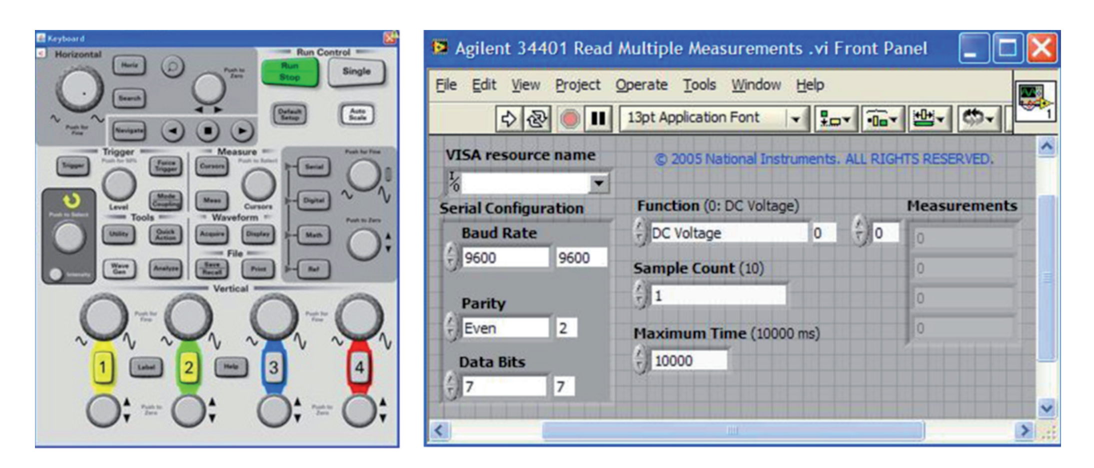


Figura 18. Ejemplo de control remoto por interfaz gráfica (LabVIEW©)

Analizando el coste de los equipos utilizados, el precio medio es casi tres veces mayor que el de un sistema de test manual. Este notable incremento es debido, mayoritariamente, a que las herramientas empleadas poseen mejores especificaciones técnicas y un potente interfaz de comunicaciones entre el dispositivo y el ordenador. Dicha interfaz de comunicaciones puede ser

implementada utilizando distintos tipos de Bus: PCI, PCI Express, PXI, PXI Express, GPIB, USB, Ethernet/LXI, etc.

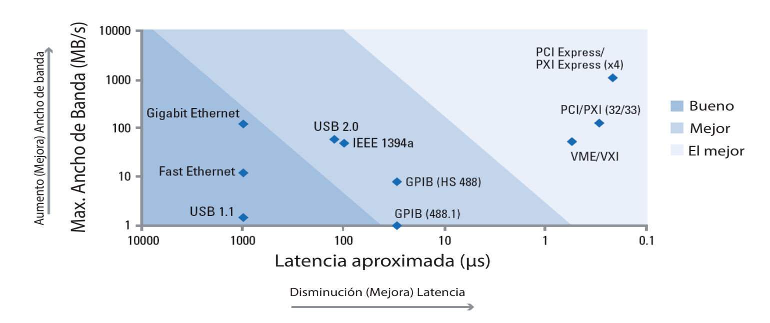


Figura 19. Comparación entre los distintos tipos de Bus [7]

4. Sistema de test automático



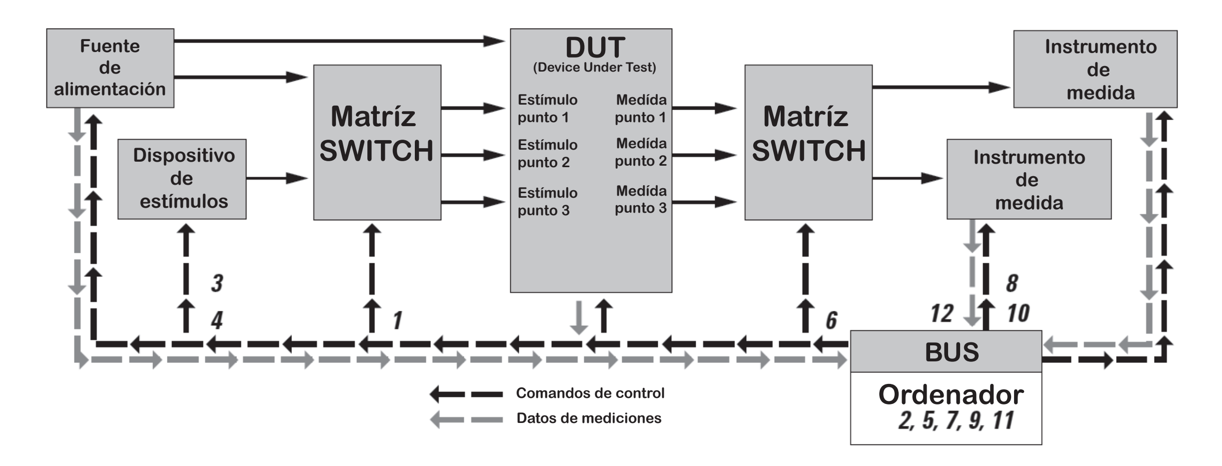
Figura 20. Entorno de trabajo de un sistema de test automático [6]

Este tipo de sistemas de test representa el último estadio de nuestro proyecto. La ejecución de cada una de las pruebas de test se desarrolla de forma plenamente autónoma.

Las razones por las que se suele emplear este tipo de entornos de trabajo serán las siguientes:

- Realización de pruebas reproducibles y repetibles.
- Necesidad de reducir el tiempo de prueba.
- Las especificaciones técnicas, que han de ser verificadas en cada secuencia, son conocidas previamente y permanecen estables a lo largo de todo el proyecto.
- Reducir el coste económico que conllevaría la realización de las pruebas empleando otro tipo de sistemas.
- Las pruebas podrán ser llevadas a cabo por personal no especializado.

El proceso propio de un sistema de test automático presentará un flujo de información similar al mostrado en la siguiente figura:



Pasos

1. Indicar a la Matriz dónde conectar el estímulo
2. *Esperar tiempo de establecimiento*
3. Configurar la señal estímulo enviada al DUT (parámetro, rango)
4. Enviar señal de estímulo
5. *Esperar tiempo de establecimiento*
6. Configurar Matriz para la medición
7. *Esperar tiempo de establecimiento*
8. Configurar instrumento para la medición
9. *Esperar tiempo de configuración del instrumento*
10. Enviar señal de medición
11. *Esperar tiempo de medición*
12. Transferir datos de la medición

Figura 21. Diagrama de flujo de un sistema de test automático [6]

El precio medio de un entorno de test completamente automático varía enormemente en función de la complejidad del dispositivo que se quiera testear, desde varios miles hasta centenas de miles de euros. Los elementos con mayor peso dentro del coste total medio son: El dispositivo de control del sistema de test ()encargado de enviar los comandos y procesar la información de las mediciones) y la automatización y/o programación del mismo.

La automatización de cada una de las pruebas se puede llevar a cabo mediante un software cerrado, donde el operario solo pueda modificar los parámetro de entrada y de salida, o mediante un lenguaje interpretado "de scripting" que pueda ser ejecutado tanto como código cerrado, como un código totalmente escalable con capacidad de ser modificado con la posibilidad de satisfacer las necesidades particulares de cada usuario.

Para automatizar las secuencias, en este proyecto se empleará el lenguaje TCL como herramienta de comunicación entre los instrumentos que componen el sistema de test y el sistema de control.

Algunas de las razones del uso de TCL en la automatización de pruebas de verificación serán las siguientes:

- Al ser un lenguaje interpretado, no necesita ser compilado, agilizando los proceso tanto de depuración como de ejecución.
- Es de código abierto, por lo que no supone un coste adicional por licencias de uso.
- Posee gran cantidad de librerías que hacen que sea totalmente compatible con los principales protocolos de comunicación (GPIB, RS232, Ethernet, USB, ect.).
- Es totalmente compatible con las metodologías [OO] y de tiempo real.

- Es compatible con la mayoría de micro-controladores del mercado, haciendo posible su utilización en pruebas de verificación que requieran mayor capacidad de procesamiento que la que puede tener normalmente un ordenador.
- Posee su propia herramienta gráfica: “TK”, que permite crear una interfaz gráfica compatible con el comandado del propio sistema de test.

Por el contrario, las desventajas que tiene este lenguaje son:

- El sistema de depuración de errores de TCL es poco intuitivo, haciendo que algunos errores de código sean complicados de depurar.
- No está tan extendido como otros lenguajes interpretados como Phyton o Perl, haciendo que el número de librerías, aunque abundantes, sigan siendo muy limitadas en comparación con otros lenguajes.

El estudio, diseño y construcción física del propio sistema de test es, en muchos casos, un proyecto independiente; el cuál se va estructurando a medida que se van cerrando las especificaciones del dispositivo que será testeado en dicho sistema.

Capítulo 2

Buses de comunicación utilizados en el comandado remoto de instrumentos

La comunicación entre los instrumentos y la unidad de control u ordenador representa uno de los elementos con mayor criticidad de todo el sistema de test. La velocidad de transmisión de datos puede suponer un factor clave en el diseño de cada prueba, pues el tiempo de transmisión requerido para procesar grandes cantidades de información puede llegar a alargar en exceso el tiempo de ejecución prueba establecido.

Como se ha mostrado en el capítulo anterior, existe una gran variedad de buses de comunicación adaptados a cada necesidad. Para la verificación de un componente electrónico, como es el caso de un convertidor DC/DC, se emplearán buses de comunicación con una tasa de transferencia de datos que varía desde los 160 kB/s (RS232) hasta los 8 MB/s (GPIB).

1. Buses de comunicación para sistemas de test con baja o media carga de procesado

1.1. RS232 (Puerto serie)



Ancho de Banda Máx: *160 kB/s*
Distribución del Ancho de Banda: *Compartido*
Calificación del A.B. : *Baja*
Calificación Latencia: *Baja*

Otras características:

- Fue uno de los primeros buses de comunicación empleado en el control de instrumentación de forma remota.
- Su implementación en el control de equipos de medición es cada vez más reducida.
- Permite distintas velocidades de transmisión mediante la configuración del Baud-rate.
- Posibilidad de establecer conexiones "null-modem" entre dispositivos.

La aparición del cable USB como sustitutivo al RS232 ha generado que el comandado de instrumentos mediante RS232 pase a un tercer plano. Actualmente, la tecnología USB ofrece grandes prestaciones tanto técnicas (ancho de banda mejorado) como de funcionabilidad (Tecnología “Plug and Play”), que hacen que en comparación con el RS232 sea, en muchos casos, la opción más acertada.

Los casos en los que se sigue empleando el puerto serie en el comandado de instrumentos pueden ser los siguientes:

- El instrumento o el equipo de control solo admiten dicho bus de comunicación.
- La velocidad de transmisión de información no es un elemento crítico en las pruebas de test que llevan a cabo en el sistema.
- No se precisa de altas prestaciones, por lo que se valoran otras opciones más económicas (como es el caso del RS232).
- Se requiere de una comunicación robusta entre dispositivos, que soporte longitudes de hasta 15 metros de separación entre el instrumento y el sistema de control.

En este proyecto, se empleará una comunicación RS232 por el puerto serie debido a que, en el caso del osciloscopio, solo es posible el comandado del instrumento a través de dicho puerto. Al utilizar una conexión RS232, quedará limitada en gran medida la gestión de los posibles errores relacionados con el estado de la conexión del osciloscopio.

Para poder establecer una comunicación entre el sistema de control y el instrumento, es necesario que exista unanimidad en la configuración de la comunicación. En el caso de que alguna de las partes (Emisor / Receptor) esté configurada de manera distinta respecto a la otra, la comunicación entre ambos no podrá ser establecida. La configuración por defecto que se utilizará será la siguiente:

- **Velocidad de transmisión:** 19200 baudios.
- **Control de flujo de datos:** no.
- **Paridad:** Impar.

Dicha configuración podrá cambiarse en cualquier momento, pero siempre deberá de llevarse a cabo de forma simultánea tanto en el equipo de control como en el instrumento.

1.2. GPIB (GPIB-USB)

Actualmente, es el bus de comunicación por excelencia, siendo uno de los más utilizados en el comandado de instrumentos.

Presenta una gran escalabilidad, permitiendo una conexión en paralelo de hasta 15 instrumentos de forma simultánea. Cada instrumento poseerá una dirección única para cada red, de forma que puedan ser comandados varios dispositivos sin que se lleguen a producir conflictos de dirección.

Algunas de las principales características de este bus de comunicación son:



Ancho de Banda Máx: *8 MB/s* **Distribución del Ancho de Banda:** *Compartido* **Calificación del AB:** *Bueno* **Calificación Latencia:** *Mejor*

Otras características:

- Gran robustez.
- Implementación extendida en la industria.
- Conexión más utilizada en instrumentos electrónicos.

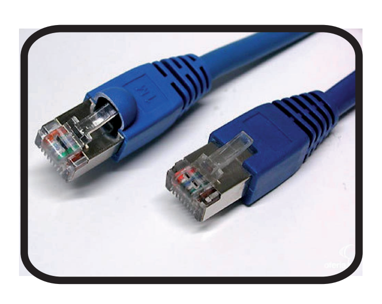
- Ideal para:
  - Automatización de equipos ya existentes.
  - Sistemas híbridos.
  - Sistemas que utilizan una instrumentación especializada.

A pesar de ser numerosas las ventajas que ofrece este bus de comunicación frente al RS232, en ciertos casos, el RS232 sigue llevando la ventaja (como por ejemplo la longitud máxima de conexión, la cual no puede ser superior a los 4 metros en el caso del GPIB frente a una longitud máxima de hasta 15 metros que soporta el RS232).

1.3. Buses de comunicación frecuentemente utilizados

Es frecuente poder encontrar otros instrumentos comandados a partir de las siguientes conexiones:

1.3.1. Ethernet



Ancho de Banda Máx: *12.5 MB/s (Fast Ethernet)* **Distribución del Ancho de Banda:** *Compartido a lo largo de la red* **Calificación del AB:** *Mejor* **Calificación Latencia:** *Bueno*

Otras características:

- Capacidad de enviar información de manera remota.
- Presente en PC's

- Ideal para:
  - Sistemas distribuidos (distanciados uno del otro).
  - Monitoreo remoto.

1.3.2. USB



Ancho de Banda Máx: *60 MB/s (Hi-Speed USB)*
Distribución del Ancho de Banda: *Compartido en los puertos*
Calificación del AB: *Bueno*
Calificación Latencia: *Mejor*

Otras características:

- Presente en todos los PC's.
- Conectividad más sencilla mediante auto--detección.

- Ideal para:
  - Aplicaciones portátiles y de escritorio.
  - Sistemas pequeños y de bajo costo.

1.4. Comparación de características

A la hora de escoger qué tipo de bus es el que mejor se adapta a los requisitos del proyecto, es frecuente recurrir a las características mostradas en la siguiente tabla:

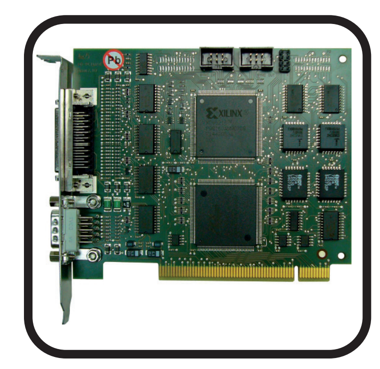
	RS232	GPIB (GPIB-USB)	USB (HUB USB)	Ethernet (switches)
Número máximo de dispositivos por puerto	1	15	127	Sin límite
Ancho de banda	20 kB/s	8MB/s	60 MB/s	12.5 MB/s
Longitud de conexión máxima	15 m	4 m	5 m	100 m
Número de dispositivos con conexión compatible	Escaso	Muy alto	Muy alto	Alto

Tabla 3. Comparativa de las principales características de distintos buses

2. Buses de comunicación para sistemas de test de alto nivel de procesado

En aquellos sistemas donde la velocidad de procesamiento requerida exceda de las especificaciones propias de un sistema de test estándar, se podrán emplear los siguiente buses de comunicación diseñados especialmente para sistemas de test de alto desempeño.

2.1. PCI y PCI Express



Ancho de Banda Máx: *250 MB/s (PCI EXPRESS) 132 MB/s (PCI)*
Distribución del Ancho de Banda: *Compartido*
Calificación del A.B. : *el mejor*
Calificación Latencia: *el mejor*

Otras características:

- Posee las mejores prestaciones tanto en ancho de banda como en latencia.
- Los ordenadores actuales cuentan la mayoría con slots PCI.

Especialmente diseñado para:

- Sistemas de trabajo de alto desempeño (S.T.A.D.)
- Procesar gran cantidad de datos

2.2. PXI y PXI Express



Ancho de Banda Máx: *250 MB/s (PXI EXPRESS) 132 MB/s (PXI)*
Distribución del Ancho de Banda: *Compartido*
Calificación del A.B. : *el mejor*
Calificación Latencia: *el mejor*

Otras características:

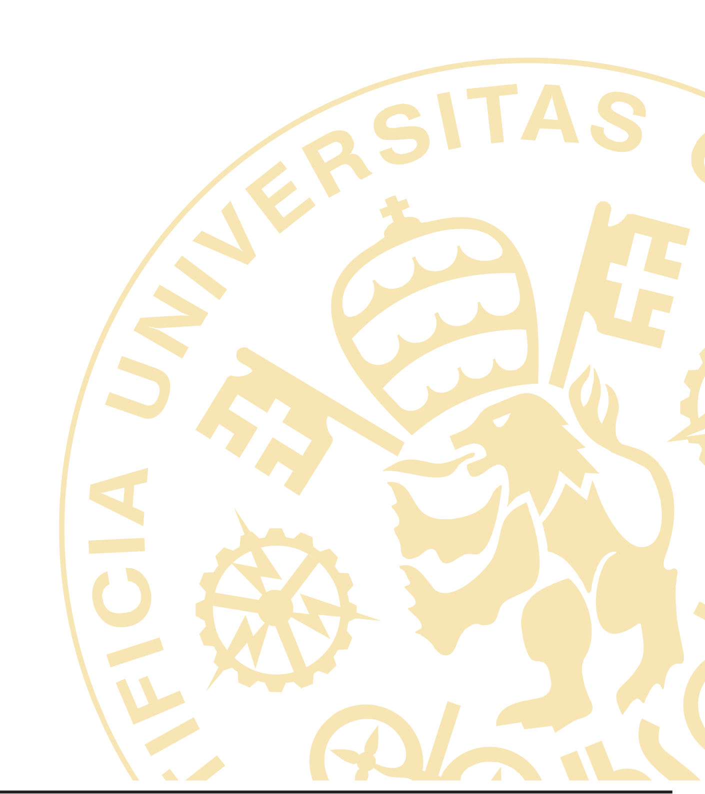
- Presenta una arquitectura física basada en CompactPCI
- Presenta funciones adicionales respecto a CompactPCI basadas en el control de tiempos y modos de sincronización.

Especialmente diseñado para:

- Sistemas de trabajo de alto desempeño (S.T.A.D.)
- Sistemas compuestos de instrumentos heterogéneos.
- Control preciso de tiempos y modos de sincronización.

PARTE III

LIBRERÍAS DEL SISTEMA



Capítulo 1

Librerías. Estructuras básicas

Cada una de las librerías de comandado representan, en su conjunto, la columna vertebral de este proyecto. El objetivo que se persigue con cada librería no es simplemente comandar un instrumento, sino comandarlo de forma controlada, segura... donde cada comando enviado al instrumento quede registrado y cada error, reportado por el instrumento, o evento ocurrido durante la secuencia de test sea gestionado de forma totalmente controlada.

Para lograr tales resultados, es necesario la combinación de distintas librerías que se encarguen de cada función por separado (comandado del instrumento, registro de errores y eventos ocurridos, gestión y ejecución de protocolos específicos en caso de registrar un error o una situación crítica para el sistema, etc.).

1. Estructura de una clase

Cada una de las librerías de comandado presentará una estructura similar entre ellas. El objetivo es establecer un modelo de librería que sea lo más escalable posible; pudiendo aplicarlo de esta forma, a cualquier instrumento que comparta un mismo protocolo de comunicación.

Dentro de la estructura de una clase, se destacarán cuatro elementos fundamentales:

- Variables: Pueden ser públicas (si se puede acceder a ellas desde fuera de la clase) o privadas (solamente la clase tiene acceso a ellas).
- Constructor: Recoge todas aquellas operaciones o acciones que han de ejecutarse en el momento en el que la clase es instanciada. Puede tener o no parámetros de entrada, necesarios para poder instanciar el objeto de dicha clase.
- Destructor: Cuando un objeto deja de estar instanciado, se ejecuta la lógica programada en el destructor de la clase. No todas las clases requieren el uso de un destructor.
- Funciones: Pueden estar orientadas al comandado del instrumento o simplemente a la ejecución de operaciones. Pueden tener, al igual que el constructor, unos parámetros de entrada requeridos por la función. Las funciones pueden ser públicas, en el caso de que puedan ser utilizadas por el usuario, o privadas, en el caso de que solo puedan ser utilizadas por la propia clase.

Cada una de las librerías presentará la siguiente estructura:

- Declaración de la librería/clase.
- Librerías adicionales [OO],abstract_class, etc.
- Estructura de la clase/librería.
- Variables privadas y públicas de la clase.
- Funciones genéricas de la clase.
- Constructor y destructor de la clase.
- Funciones de la clase.

```
#Se nombra la libreria del instrumento:
package provide nombre_libreria 1.0
#Se inicializa la clase abstracta correspondiente al instrumento:
package require nombre_clase_abstracta
#Se inicializan librerias adicionales si es necesario:
package require nombre_libreria_adicional
#Se inicializa la libreria que permite instanciar objetos en TCL:
package require Itcl
#Se declara el namespace donde estara instanciada la libreria:
namespace eval nombre_librería {

      #Se importa el namespace correspondiente a las librerias adicionales
      namespace import ::nombre_libreria_adicional::*
      namespace import ::itcl::*

      #Se declara la clase del instrumento:
      ::itcl::class nombre_instrumento {

        #Se importa la clase abstracta correspondiente al tipo de instrumento
        inherit ::class_instrumento::instrumento

        #Se declaran las variables que seran utilizadas en la clase
        private variable a
        private variable b
        public  variable c

        ##################################################################################################
        #Funiones genericas de la clase:
        ##################################################################################################

        proc sumar_ab {a b} {
            set a_mas_b [expr $a+$b]
            return $a_mas_b
        }

        ##################################################################################################
        #Funiones del constructor y destructor pertenecientes a un instrumento:
        ##################################################################################################

        #Se instancia el constructor de la clase
        constructor {parametro_1 parametro_2} {
          #EJEMPLO DE FUNCIÓN DEL CONSTRUCTOR:
            #Se abre el canal de comunicacion del instrumento.
            #Se instancia el diccionario de errores (Si lo tiene).
            #Se configura la severidad correspondiente a un error de conexión del instrumento.
            #Se añade la clase del instrumento a la clase "error_handle" encargada de gestionar situaciones criticas.
            #Se inicializa el valor de severidad actual a 0.
        }


        destructor {parametro_1 parametro_2} {
          #EJEMPLO DE FUNCIÓN  DEL DESTRUCTOR:
            #Se cierra el canal de comunicacion del instrumento.
        }

        ##################################################################################################
        #Ejemplo de funciones que componen una clase:
        ##################################################################################################

        #Se declaran las funciones publicas que componen la clase:
        public method funcion_1 { parametro_1 parametro_2} {
           #Conjunto de comandos correspondientes a la funcion_1
           set a $parametro_1
           set b $parametro_2

           #Para utilizar funciones privadas dentro de una clase, se hara de la siguiente forma:
           set c [$this funcion_privada $a $b]

           return $c

        }
```

```
        #Se declaran las funciones privadas que podran ser solamente usadas dentro de la clase
        private method funcion_privada { parametro_a parametro_b } {
            set valor [expr $parametro_a + $parametro_b]
            set valor [sumar_ab $parametro_a $parametro_b]
            return $valor
        }
    }

    #Se exporta la clase correspondiente al instrumento
    namespace export nombre_libreria
}
```

Código 1. Ejemplo de estructura básica de una clase

2. Estructura del constructor. Instrumentos comandados por GPIB

El constructor es la "función" a la que se llama automáticamente siempre que un objeto de esta clase sea instanciado. Un constructor puede requerir unos argumentos de entrada. Dichos argumentos serán necesarios para poder instanciar el objeto.

De manera común para todos los instrumentos que utilizan un protocolo de comunicación GPIB | IEEE-488, el constructor correspondiente a su clase presentará la siguiente estructura:

Dirección GPIB: `_visa_addrs` — Objeto driver- gpib: `_gpib` — Objeto error_handle: `d_error_handle`

```
constructor { _visa_addrs  _gpib  d_error_handle} {

  #Pointer to $$error_handle class
  set error_handle $d_error_handle
  #Pointer to driver_gpib
  set gpib $_gpib
```

Se asignan a una variable los objetos de la clase *driver_gpib* y *error_handle*

```
  # open device
  set visaAddr $_visa_addrs
  # get handle to default resource manager
  if { [catch { set rm [visa::open-default-rm] } rc] } {
      puts stderr "Error opening default resource manager\n$rc"
  } else {
      set rm [visa::open-default-rm]
  }

  # check if devide opened
  if { [catch { set vi [visa::open $rm $visaAddr] } rc] } {
      puts "Error opening instrument `$visaAddr`\n$rc"
  } else {
      set vi [visa::open $rm "$visaAddr"]
      # Set proper timeout
      fconfigure $vi -timeout 500
  }
```

Se configura la comunicación con el instrumento por el canal GPIB previamente indicado al inicializar el objeto correspondiente al instrumento. En el caso de que no se pueda establecer una comunicación con el instrumento, se mostrará un error por pantalla.

```
  # Get ID from instrument
  puts $vi "*IDN?"
  #Remove useless part of ID
  set _id [gets $vi]
  #Characters until second ","
  set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
  #Redefine device ID
  set id [string range $_id 0 $num]
  #Device name
  set device_name $id
```

Se guarda la identificación [nombre y número de serie] del instrumento para ser utilizada posteriormente en el archivo de registro con todos los comandos mandados al instrumento, incluyendo la fecha y la hora.

```
#Setup error dictionary
set fp [open [file join  setup sw source tcl driver_gpib_hp6653a
driver_gpib_hp6653a_dict_error.json] r]
set dict_err [read $fp]
set dict_err [json::json2dict $dict_err]
```

Se instancia el diccionario de errores del dispositivo asignándolo a una variable en formato de diccionario, para lo cual se requiere la conversión de formato JSON a TCL.

```
# SEVERITY CONNECTION ERROR VALUE
set severity_nu 2
set connection_error_value -1
set c_err_msg [list "$connection_error_value, \"Connection error\" $severity_nu"
                    "+0, \"No Error\" 0"]
set con_def_msg "error writing \" $vi\": Unknown error"
```

Se asigna al error de desconexión un número de error (-1) y un nivel de severidad específico (2). Dichos valores pueden cambiar de un instrumento a otro.

```
#Specific error report when disconection
set c_err_msg_er "$connection_error_value, \"Connection error\" $severity_nu"
```

Se define el mensaje que se mostrará por pantalla al producirse una desconexión.

```
#Add instrument in error_handle
$this setup_error_handle
```

Se llama al método "setup_error_handle" el cual añade la referencia de este objeto al objeto error_handle.

```
#Initialize maximum severity
set max 0
}
```

Inicia el nivel de severidad a 0.

Código 2. Estructura básica de un constructor para instrumentos comandados por GPIB

3. Estructura del constructor y destructor. Instrumentos comandados por RS232

El constructor de la clase de un instrumento comandado por RS232 presentará una estructura muy simple, debido a que muchas de las funciones implementadas en GPIB (gestión de errores, detección de instrumentos, etc.) son altamente ineficientes en comunicaciones RS232.

Dichas limitaciones vienen determinadas por la forma con la que se debe gestionar el canal de comunicación. Para poder comunicarse con un instrumento, se deberá abrir un canal en el puerto serie (COM1, COM2,..) que permanecerá abierto hasta que se cierre de forma manual. Esto imposibilita el poder detectar la desconexión del instrumento; ya que al preguntar el estado en el que se encuentra, no se recibirá respuesta alguna. Esta situación podría interpretarse como una pérdida de conexión o simplemente que el instrumento está en ese momento ocupado.

La forma más efectiva de lidiar con situaciones en las que se desconoce el estado del instrumento suele ser, generalmente, un contador de bits combinado con un "time-out" efectivo.

La configuración del "time-out" hace referencia al tiempo (ms) que el sistema esperará a recibir una respuesta por parte del instrumento. En el caso en que el sistema intente leer del buffer de memoria un número determinado de bits y el instrumento no sea capaz de devolver tal cantidad, esperará el tiempo indicado en el "time-out" antes de interrumpir el proceso de lectura de bits del buffer. De este modo se evita que el sistema se colapse esperando una cadena de bits inexistente.

El constructor de un instrumento comandado por RS232 presentará los siguientes elementos:


```
constructor { serial_port _config_dir } {

    set vi [open $serial_port: RDWR]

    fconfigure $vi -blocking 1
    fconfigure $vi -buffering full
    fconfigure $vi -encoding binary
    fconfigure $vi -mode 19200,o,8,1
    fconfigure $vi -translation binary
    fconfigure $vi -eofchar {}
    fconfigure $vi -timeout 10000

    #Configuration files location
    set config_dir $_config_dir
}
```

Código 3. Estructura básica de un constructor para instrumentos comandados por RS232

En comunicaciones RS232 también será necesario la implementación de un destructor. El destructor contiene las acciones que serán ejecutadas en el momento en el que el objeto deje de estar instanciado (Fin de la prueba).

Al contrario que el constructor, nunca requiere de parámetros de entrada. La función del destructor en comunicaciones RS232 será simplemente cerrar el canal de comunicaciones. Su estructura será la siguiente:

```
# Destructor of the class
destructor {
    close $vi
}
```


Código 4. Estructura básica de un destructor para instrumentos comandados por RS232

4. Estructura de una función de comandado

Las funciones de comandado de cada uno de los instrumentos, a pesar de ser funciones totalmente distintas, presentan una estructura común a todas.

Además de establecer un flujo de comandos entre el ordenador y el instrumento, cada una de las funciones debe cumplir con las siguientes especificaciones:

- **Comandar el instrumento:** Poder configurar y controlar el instrumento de forma remota. Es el principal objetivo de cada una de las funciones. Es bastante frecuente que para una única función se requieran más de un solo comando, por lo que es de vital importancia el orden de cada uno de ellos. Uno de los errores más comunes tiene su origen en el orden con el que el buffer de memoria es vaciado. Un vaciado incorrecto puede producir que la medida obtenida sea incoherente o simplemente, si el buffer se encontraba vacío, no exista.

- **Registrar cada uno de los comandos enviados desde el sistema de control:** A la hora de depurar una secuencia de test, es muy difícil poder realizar un seguimiento en detalle de la misma si no se conoce qué comandos se están enviando en todo momento. El registro debe contener: la hora en la que fue enviado, la dirección del instrumento y el comando enviado.

- **Detectar cualquier error producido:** La aparición de errores durante la ejecución de una secuencia es un suceso que debe de evitarse a toda costa, pero no siempre es posible. Durante la ejecución de la secuencia pueden producirse diferentes tipos de errores:

  - **Errores de comandado:** Suelen aparecer cuando se introduce un comando con una sintaxis correcta, pero las unidades empleadas no son las correctas, la cifra es excesivamente alta, etc. El sistema por sí solo no es capaz de detectarlos, sino que es el instrumento quien los devuelve.
  - **Errores de conexión:** Ocurren al intentar mandar un comando a un instrumento que ha perdido la comunicación con el equipo de control. Este tipo de errores solo pueden ser detectados por el propio sistema de control.
  - **Errores de sintaxis:** Aparecen al cometer un error de sintaxis en el código (una variable mal declarada, un paréntesis mal localizado...). Son errores frecuentes de sintaxis.

- **Registrar la severidad asociada a un error:** Uno de los elementos más importantes de cada una de las pruebas de test es la parada de emergencia controlada. La ejecución de la misma depende de la severidad del error. Dicha parada de emergencia es gestionada por el objeto de la clase *error_handle*. Para ello, es necesario que cada una de las funciones de comandado registre la severidad de cada error y que sea capaz de enviar al gestor de errores dicha información.

Utilizar una misma estructura que satisfaga todos los requisitos previamente descritos permite:

- Poder abstraer cada una de las funciones. Abstrayendo las funciones se consigue una mayor escalabilidad del sistema. Para añadir más funciones de comandado solamente es necesario diseñar la parte lógica de la función, pues el resto de elementos son totalmente compatibles con cualquier función.

- Mayor compatibilidad entre elementos. Al utilizar la misma estructura, se asegura una plena compatibilidad entre funciones.

- Depurado de errores más efectivo. Al compartir un mismo esquema de trabajo, cualquier error detectado en una función puede ser solventado en el resto sin preocuparse por la estructura interna de cada función.

La estructura básica de una función de comandado de instrumentos presenta los siguientes elementos:

Parámetro de entrada

```
public method read_voltage_dc { v_conf_dc } {
#================================================================================
if { [catch {
#================================================================================
    array set v_param $v_conf_dc
    puts $vi "CONF:VOLT:DC $v_param(range), $v_param(resolution)"

    $gpib command_sent_wfile $device_name $visaAddr "CONF:VOLT:DC  \
                             $v_param(range), $v_param(resolution)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    puts $vi "READ?"
    set id [gets $vi]

    $gpib command_sent_wfile $device_name $visaAddr "READ?"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
#================================================================================
} err_code]} {
    if {$err_code == $con_def_msg} {
        $error_handle error_report_screen $visaAddr $c_err_msg
    } else {
        $error_handle error_report_screen $visaAddr "Unknown error"
    }
}
#================================================================================
$error_handle  execute_controlled_exit [$this error_report severity]

if { $err_code == "" } {
#================================================================================
    return $id
#================================================================================
}
}
```

Estructura común: `if { [catch {`

Función encapsulada:
- Envío de comandos al instrumento.
- Se registran los comandos enviados y se comprueba que no haya errores.
- Se lee del buffer de memoria las posibles medidas obtenidas del instrumento
- Se vuelven a registran los comandos enviados y se comprueba que no haya errores.

Estructura común:
- Detección de errores de comunicación.
- Detección de errores de sintaxis | otros errores.
- Se envía al gestor de errores el nivel de severidad máximo producido

Estructura común:
- Si no se ha producido ningún error se devuelve el valor medido.

Código 5. Estructura básica de una función de comandado

5. Estructura de un diccionario de errores

Cada uno de los instrumentos que componen el sistema de test posee su propia sintaxis de errores. Con el objetivo de poder interpretar tanto un error numérico como un mensaje, utilizaremos un diccionario propio, creado a partir del manual de usuario del instrumento.

El diccionario estará compuesto por:

- Código de error: Valor numérico correspondiente a un error, que será interpretado como un mensaje.
- Mensaje de error: Descripción asociada al valor numérico devuelto por el instrumento.
- Severidad: Valor numérico que hace referencia al nivel de criticidad del error. Dependiendo de dicho valor, se llevará acabo un protocolo de seguridad asignado a dicho valor.

Cada diccionario estará estructurado como un *array* siguiendo la nomenclatura JSON (*JavaScript Object Notation*). La razón de su utilización recae en su compatibilidad con otros muchos lenguajes de programación, permitiendo así, el poder ser utilizado en cualquiera de ellos.

Un diccionario de errores escrito en JSON presenta la siguiente estructura:

```
{
"Error_value":{"Error_Message":"Severity_value"},
"+0":   {"No error"                        :"0"},
"-101": {"Invalid character"               :"1"},
"-102": {"Syntax error"                    :"1"},
"-113": {"Undefined header"                :"1"},
"-121": {"Invalid character in number"     :"1"},
"-123": {"Numeric overflow"                :"2"},
                 ***
"-214": {"Trigger deadlock"                :"2"},
"-221": {"Settings conflict"               :"3"},
"-222": {"Data out of range"               :"1"},
"-223": {"Too much data"                   :"1"},
"-224": {"Illegal parameter value"         :"1"},
"-230": {"Data stale"                      :"1"},
"-330": {"Self-test failed"                :"4"},
"-350": {"Too many errors"                 :"5"}
}
```

Código 6. Estructura básica de un diccionario de errores en formato JSON

6. Estructura del Setup de secuencia

Cada secuencia de test requiere de un Setup específico que inicialice todos los instrumentos, funciones, librerías, etc. que se necesiten para ejecutar dicha secuencia.

Las razones por las que se emplea un Setup como método de inicialización del sistema son las siguientes:

- Permite configurar cada secuencia individualmente, eligiendo qué instrumentos y qué funciones forman parte de la misma.
- Permite establecer una comunicación entre las funciones del sistema y los instrumentos. Dicha comunicación posibilita: registrar todos los comandos enviados, gestionar una salida de secuencia controlada en caso de error, etc.

Un Setup de secuencia presentará una estructura similar a la siguiente:

```
#Se nombra la libreria del setup de secuencia:
package provide setup_secuencia_ejemplo 1.0

#Se inicializan las librerias correspondientes a las funciones del sistema
package require funcion_error_handle
package require funcion_comunicacion_busGPIB

#Se inicializan las librerias correspondientes a los instrumentos y sus clases abstractas
package require class_fuente_alimentacion
package require driver_gpib_hp6653a

#Se importa el namespace de cada libreria
namespace import ::funcion_comunicacion_busGPIB::*
namespace import ::funcion_error_handle::*
namespace import ::class_fuente_alimentacion::*
namespace import ::driver_gpib_hp6653a::*

#Se instancian las funciones logicas del sistema:

#Error Handle: Encargado de gestionar las salidas de secuencia controladas
set e_h [error_handle e_h]

#Driver GPIB: encargado de gestionar y registrar la comunicacion entre el ordenador y el instrumento
set d_gpib [driver_gpib d_gpib]

#Se instancia el instrumento que sera comandado en la secuencia

set PS_main [driver_gpib_hp6653a PS_main "GPIB0::5::INSTR" $d_gpib $e_h]
```

Declaración de la librería Setup.
Inicialización de funciones e instrumentos.
Instanciación de funciones del sistema.
Instanciación de instrumentos.

*[El instrumento **PS_main** podrá utilizar funciones de **e_h** y **d_gpib**]*

Código 7. Ejemplo de estructura básica de un Setup de secuencia

7. Lista de instrumentos de secuencia

En el setup de la secuencia es necesario crear una lista con todos los instrumentos que van a participar en la prueba de test. Esta lista será utilizada por el objeto perteneciente a la clase *driver_gpib* para poder comprobar la correcta conexión de todos los instrumentos que van a participar en la prueba. Su función se explicará más adelante.

La lista de instrumentos deberá tener la siguiente estructura:

- Primer elemento: Contiene la dirección del instrumento.
- Segundo elemento: Contiene el nombre del instrumento.
- Tercer elemento: Contiene el protocolo de comunicación empleado.

Ejemplo de lista de instrumentos:

```
set list_devices [list                                                        \
{visa_addrs GPIB0::2::INSTR name HEWLETT-PACKARD,34401A protocol gpib} \
{visa_addrs GPIB0::5::INSTR name HEWLETT-PA CKA,RD6653A protocol gpib} \
{visa_addrs GPIB0::1::INSTR name KIKUSUI,PLZ-50F        protocol gpib} ]
```

Código 8. Ejemplo de lista de instrumentos

8. Instanciación de clases (Setup)

Para poder comandar un instrumento, primero se deberá instanciar un objeto perteneciente a la clase del instrumento. Antes de instanciarlo, es necesario importar el resto de clases y opcionalmente, también aquellos "namespaces" donde se encuentre declarada la clase.

El siguiente fragmento de código muestra dos posibles formas de instanciar un objeto en TCL:

```
package require "package_que_contiene_la_clase"
namespace import "::namespace_de_la_clase::*"

set nombre_puntero_obj [nombre_clase  nombre_objeto  $param_1  $param_2]

==============================================================

package require "package_que_contiene_la_clase"

set nombre_puntero_obj [::namespace_de_la_clase::nombre_clase  nombre_objeto  $param_1  $param_2 ]
```

Código 9. Ejemplo de instanciación de un objeto

La principal diferencia entre ambas es la forma de referenciar la clase del objeto. Si se importa el espacio de nombres donde está contenida la clase, no será necesario indicar su dirección completa porque ya estará contenida en el "namespace" actual.

PARTE IV

SECUENCIA DE TEST Y RESULTADOS OBTENIDOS



Motivación

Para poder apreciar las grandes ventajas que ofrece un sistema de test totalmente automatizado, se llevará a cabo la verificación del convertidor previamente descrito.

Capítulo 1

Prueba de verificación. Diseño del setup

Para comenzar a utilizar los instrumentos es necesario instanciarlo. Se diseñará un fichero de Setup que permita instanciar los instrumentos de manera rápida y eficiente. Para poder definir las partes principales del setup, lo dividiremos en dos:

1. Instanciación de librerías y espacios de nombres

```
package provide setup_prueba_convertidor 1.0

#==================================================
#===========INSTANCIACION DE PAQUETES==============
#==================================================

#INSTRUMETOS
package require driver_gpib_hp34401a;
package require driver_gpib_hp6653a
package require driver_gpib_kikuplz150u
package require driver_rs232_tekttps2024

#CLASES ABSTRACTAS DE CADA INSTRUMENTO
package require class_multimeter
package require class_powersupply
package require class_electronic_load

#REPORTING Y GESTION DE ERRORES
package require driver_gpib
package require error_handle

#OTRAS LIBRERIAS
package require json
package require Itcl
package require common
package require logger
```

```
#==================================================
#===========INSTANCIACION DE NAMESPACES===========
#==================================================

#INSTRUMETOS
namespace import ::driver_gpib_hp34401a::*
namespace import ::driver_gpib_hp6653a::*
namespace import ::driver_gpib_kikuplz150u::*
namespace import ::driver_rs232_tekttps2024::*

#CLASES ABSTRACTAS DE CADA INSTRUMENTO
namespace import ::class_multimeter::*
namespace import ::class_powersupply::*
namespace import ::class_electronic_load::*

#REPORTING Y GESTION DE ERRORES
namespace import ::driver_gpib::*
namespace import ::error_handle::*

#OTRAS LIBRERIAS
namespace import ::common::*
namespace import ::logger::*
```

Código 10. Instanciación de librerías y espacios de nombres

2. Instanciación de objetos (Instrumentos y funciones)

```
==========================================================================================
# SETUP CRLogger (Configura la forma con que se muestran los mensajes por pantalla):
set log [::logger::CRLogger]
==========================================================================================
# Lista de instrumentos comandados por GPIB que van a particpar en la prueba de test:
set list_devices [list \
{visa_addrs   GPIB0::2::INSTR     name      HEWLETT-PACKARD,34401A      protocol     gpib} \
{visa_addrs   GPIB0::5::INSTR     name      HEWLETT-PACKARD, 6653A      protocol     gpib} \
{visa_addrs   GPIB0::1::INSTR     name      KIKUSUI,PLZ-50F             protocol     gpib} ]

# {Directorio del fichero con todos los comandos enviados} {nombre del fichero }
set driver_log_name [ list \
 {work/test/board_characterization_mst1553std_ssb/seq10001_convertidor/commands_sent} \
 {test1_log} ]

# SETUP driver_gpib
set d_gpib [driver_gpib d_gpib $driver_log_name $inlist]
==========================================================================================
# SETUP error_handle
set e_h [error_handle e_h $log]
==========================================================================================
# SETUP FUENTE DE ALIMENTACION:
set PS_main [driver_gpib_hp6653a PS_main "GPIB0::5::INSTR" $d_gpib $e_h]
# SETUP CARGA DINAMICA:
set DL_test [driver_gpib_kikuplz150u DL_test "GPIB0::1::INSTR" $d_gpib $e_h]
# SETUP MULTIMETRO
set Mult_test [driver_gpib_hp34401a Mult_test "GPIB0::2::INSTR" $d_gpib $e_h]
==========================================================================================
# DIRECTORIO DONDE SE ENCUENTRAN LOS FICHEROS DE CONFIGURACION DEL OSCILOSCOPIO
set os_config_dir "/setup_prueba_convertidor/"

# SETUP OSCILOSCOPIO
set Tek [driver_rs232_tekttps2024 tek "COM1" $os_config_dir]
```

Código 11. Instanciación de objetos (Instrumentos y funciones)

3. Arranque del Setup

Para cargar el setup de la prueba, simplemente se deberá ejecutar el siguiente comando:

```
source [file setup_prueba_convertidor.tcl]
```

Capítulo 2

Prueba de verificación. Diseño de la secuencia

El fichero de secuencia representa la parte lógica de cada una de las pruebas de test. En dichas pruebas es muy importante considerar el mayor número posible de situaciones que puedan darse a lo largo de la secuencia: Un corto circuito, sobrecarga del sistema, desconexión de un instrumento, etc. Cuanto mayor sea el número de situaciones consideradas, más compleja será la lógica de comandado.

La estructura de la secuencia estará dividida en cuatro partes principales:

1. Cabecera

La primera parte de la secuencia será la cabecera, donde quedará reflejada la identificación de la prueba que se va a llevar a cabo. La finalidad de esta parte es meramente documental.

```
#================================================================================
# sequence definition
#================================================================================
set seqdef(name)       [get_sequence_name]
set seqdef(dut)        "Convertidor_DC_DC"
set seqdef(testbench)  "hardware_testbench"
set seqdef(objective)  "Verificar especificaciones técnicas del convertidor"
set seqdef(related)    ""
set seqdef(logger)     $log
set seqdef(simstep)    /tb_system/simstep
set seq "[namespace current]::[sequence seq [array get seqdef]]"

set step_id -10

#================================================================================
#                        S T E P   D E F I N I T I O N"
#================================================================================
set step [seq new_step [list id [incr step_id 10] objective "Tes0" \
   requirements "" meth "automatic" on_exit "\$this print_result"]]
#================================================================================
```

Código 12. Cabecera de la secuencia

2. Parámetros de configuración de la prueba

Para conseguir que cada prueba pueda ser fácilmente configurable, es necesario que los parámetros de configuración sean lo más accesibles posible. Por ello, todos los parámetros serán declarados al principio de la secuencia.

```
#====================================================================================
#              P A R A M E T R O S    D E    C O N F I G U R A C I O N
#====================================================================================

#VALOR DE RESISTENCIA DE LA CARGA DINAMICA DURANTE EL INCREMENTO Y REDUCCION DE TENSION
set dl_resistance_cte {100}; #Ohms

#VALORES DE RESISTENCIA DE LA CARGA DINAMICA DURANTE EL PERIODO DE TENSION CONSTANTE
set dl_resistance_secquence {2000 1000 500 100 50}

#TENSION INICIAL DE LA SECUENCIA
set vmin_up {8};# V

#TENSION FINAL DE LA SECUENCIA
set vmin_down {4};# V

#NIVEL DE TENSION CONSTANTE
set vmax {24};# V

#INCREMENTO DE TENSION POR CADA STEP DE SECUENCIA
set v_step {1};# V

#TIEMPO DE ESTABILIZACION DE MEDIDAS
set tm {1000}; # ms

#VALOR DE CORRIENTE POR ENCIMA DEL CUAL SE CONSIDERA QUE EL CONVERTIDOR ESTA ENCENDIDO
set ion {0.005}; #A

#VALOR DE CORRIENTE POR DEBAJO DEL CUAL SE CONSIDERA QUE EL CONVERTIDOR ESTA APAGADO
set ioff {0.010}; #A

#CORRIENTE MAXIMA DE SALIDA DE LA FUENTE DE ALIMENTACION
set imax {200 mA}

#PARAMETROS DE PROTECCION DE LA FUENTE DE ALIMENTACION
#NIVEL DE TENSION DE PROTECCION
set v_prot {30 V}

#NIVEL DE CORRIENTE DE PROTECCION
set i_prot {300 mA}


#MODULO DE LA CARGA DINAMICA AL QUE ESTA CONECTADO LA SALIDA DEL CONVERTIDOR
set dl_chan_resist "ch1"
```

Código 13. Parámetros de configuración de la secuencia

3. Apagado controlado. Interrupción de secuencia

En el caso de que se produzca un error con una severidad asociada superior a la permitida, se ejecutará un apagado controlado del sistema para no provocar daños tanto al dispositivo bajo test como a los propios instrumentos.

La secuencia de parada se define después de los parámetros de configuración de secuencia. En el caso de la prueba del convertidor, el apagado controlado será el siguiente:

```
set seq_controlled_exit {
    "
    #Apagamos la fuente de alimentacion
    $PS_main output off
    #Desactivamos la fuente de alimentacion de la carga dinamica
    $DL_test output off
    #Desactivamos el consumo de potencia de la carga dinamica
    $DL_test input off
    #Desactivamos las protecciones de la fuente de alimentacion
    $PS_main clear_prot
    "
}

#Se instancia la secuencia de apagado controlado en el gestor de errores
e_h load_controlled_exit $seq_controlled_exit
```

Código 14. Secuencia de apagado controlado e instanciación de la misma en el gestor de errores

4. Procedimientos básicos

Con el objetivo de simplificar la apariencia del código, se unifican aquellas secuencias que vayan a ser instanciadas en varias ocasiones, consiguiendo de esta forma simplificar la lógica de la prueba. Para esta secuencia, se declarará, a modo de ejemplo, la función que finaliza la secuencia de test:

```
#Ejemplo de procedimiento basico: Fin de la prueba
proc end_test {} {
    #Apagado de la fuente
    $PS_main output off
    $PS_main clear_prot
    #Apagado de la carga dinamica
    $DL_test output off
    $DL_test input off
}
```

Código 15. Ejemplo de procedimiento básico. Fin de la prueba

5. Secuencia lógica de la prueba

Una vez definidos todos los parámetros de la prueba, se diseña la secuencia lógica de la misma.

5.1. Configuración inicial de instrumentos

```
#SE PRESENTAN POR PANTALLA LOS PARAMETROS DE CONFIGURACION DE LA PRUEBA:
$step add_procedure "
    Dynamic load resistance:     $dl_resistance_cte Ohms
    Minimun voltage rising:      $vmin_up V
    Minimun voltage decreasing:  $vmin_down V
    Maximun voltage:             $vmax V
    Voltage step:                $v_step V
    Current on:                  $ion A
    Current off:                 $ioff A
    Step time (setting time):    $tm ms"

#SE COMPRUEBA QUE TODOS LOS INSTRUMENTOS ESTAN CONECTADOS AL SISTEMA DE CONTROL:
if { [d_gpib check] == 0 } {
    #EN CASO DE HABER ALGUN ERROR DE COMUNICACION SE REPORTA UN ERROR
    $step  rep_error "Error: some devices are not connected"
} else {


    #===========================================================================================================
    #                       C O N F I G U R A C I O N   D E   I N S T R U M E N T O S
    #===========================================================================================================

    #SE CARGA LA CONFIGURACION DEL OSCILOSCOPIO PARA DETECTAR EL ENCENDIDO DEL CONVERTIDOR (ESCALON DE SUBIDA)
    $Tek setconfig subida
    #===========================================================================================================
    #CARGA DINAMICA EN MODO RESISTENCIA
    $DL_test set_mode [list channel $dl_chan_resist mode cr]
    $DL_test set_conductance [list channel $dl_chan_resist conductance [expr 1.0/$dl_resistance_cte] mode auto]
    #ENCENDIDO DE ALIMENTACION Y CONSUMO DE LA CARGA DINAMICA
    $DL_test input on
    $DL_test output on
    $step  add_procedure "setup DL as resistance: $dl_resistance_cte"
    #===========================================================================================================
    #CONFIGURACION DE TENSION Y CORRIENTE DE LA FUENTE DE ALIMENTACION
    $PS_main set_psv [list voltage $vmin_up]
    $PS_main set_psc [list current $imax]
    #CONFIGURACION DE PROTECCION PARA TENSION Y CORRIENTE
    $PS_main set_prot_psvc [ list v_level $v_prot   c_level $i_prot]
    #SE ACTIVA LA ALIMENTACION
    $PS_main output on
    #===========================================================================================================
    #INICIALIZACION DE LOS FLAGS DE CORRIENTE DE ENCENDIDO Y APAGADO
    # Current:
    set c_thres_on 0
    set c_thres_off 0
    #===========================================================================================================
    #SE ESTABLECE UN CANAL HACIA EL ARCHIVO DONDE ESTARAN REGISTRADAS TODAS LAS MEDIDAS
    set report_measures [file join test2_measures.txt]
    set fp [open $report_measures w]
    #===========================================================================================================
    #SE INICIALIZA LA VARIABLE step_counter
    set  step_counter 0
    #===========================================================================================================
    #SE INICIALIZAN LAS VARIABLES DE TENSION Y CORRIENTE (VALORES PREVIOS DE CADA MEDIDA)
    set v_ps_output_mult [$PS_main read_outputs {voltage}]
    set c_ps_output [$PS_main read_outputs {current}]
```

Código 16. Configuración inicial de los instrumentos

5.2. Incremento de la tensión con carga constante

```
#==========================================================================================================
#           I N C R E M E N T O   D E   L A   T E N S I O N   C O N   C A R G A   C O N S T A N T E
#==========================================================================================================

#SE INICIALIZA EL NUMERO DE INCREMENTOS DE TENSION REALIZADOS
set x 0

#CONDICION DE INCREMENTO DE TENSION: TENSION DE ENTRADA CONVERTIDOR < TENSION MAXIMA
while {[$Mult_test read_voltage_dc [list range def resolution def]] < $vmax && \
       [$PS_main read_outputs {voltage}] < $vmax {


    #SE GUARDAN LAS MEDICIONES PREVIAS DE TENSION Y CORRIENTE
    set v_prev $v_ps_output_mult
    set c_prev $c_ps_output


    #SE INCREMENTA LA TENSION DE ALIMENTACION. SI ALCANZA EL VALOR MAXIMO, LA TENSION SATURA A DICHO VALOR.
    if {  [expr ($vmin_up+$x*$v_step)] < $vmax } {
        $PS_main set_psv [list voltage [expr ($vmin_up+$x*$v_step)]A];#Mal A
    } else {
        $PS_main set_psv [list voltage $vmax]
    }


    # AUMENTA EL INCREMENTO DE TENSION
    incr x


    #ESPERAR EL TIEMPO DE ESTABILIZACION DE LAS MEDIDAS
    after $tm


    #COMPROBAR QUE LA PROTECCION DE LA FUENTE DE ALIMENTACION NO HA REPORTADO NINGUN EVENTO
    set protection_event [$PS_main read_events]


    #SI SE DETECTA UN EVENTO DE PROTECCION. SE PARA LA SECUENCIA Y SE REPORTA UN ERROR.
    if {[expr $protection_event] > 0} {
        end_test
        $step rep_error "Error test stopped due to protection event number: $protection_event"
        error "Protection event : $protection_event"
    }

    # SE ASIGNAN TODAS LAS MEDICIONES DE CADA (INSTRUMENTO) A SU VARIABLE CORRESPONDIENTE
    # TENSION DE ENTRADA (FUENTE DE ALIMENTACION)
    set v_ps_output_ps [$PS_main read_outputs {voltage}]
    # TENSION DE ENTRADA (MULTIMETRO)
    set v_ps_output_mult [$Mult_test read_voltage_dc [list range $v_ps_output_ps resolution def]]
    # CORRIENTE DE ENTRADA (FUENTE DE ALIMENTACION)
    set c_ps_output [$PS_main read_outputs {current}]
    # TENSION DE SALIDA (CARGA DINAMICA)
    set v_dl_inpunt [$DL_test read_voltage "$dl_chan_resist"]
    # CORRIENTE DE SALIDA (CARGA DINAMICA)
    set c_dl_inpunt [$DL_test read_current "$dl_chan_resist"]

    #SE MUESTRAN LOS RESULTADOS POR PANTALLA
    $step add_procedure "
    Voltage PS output measure (PS measure):          $v_ps_output_ps V
    Current PS output measure (PS measure):          $c_ps_output A
    Voltage dynamic load (Resistance mode):          $v_dl_inpunt V
    Current dynamic load (Resistance mode):          $c_dl_inpunt A
    Voltage PS output measure (multimeter measure): $v_ps_output_mult V
    "
```

```
    #COMPROBACION DE ESTADO DE ENCENDIDO DEL CONVERTIDOR: (CORRIENTE_DESPUES/CORRIENTE_ANTES) > 10
    if {[expr abs($c_ps_output)/abs($c_prev)] > $current_peak_detection_value && $c_thres_on == 0} {
        #SE PONE EL FLAG DE ENCENDIDO A 1
        set c_thres_on 1

        #SE EXPORTA LA CAPTURA DEL OSCILOSCOPIO
        $Tek hard_copy subida

        #SE GUARDA LA TENSION Y CORRIENTE DE ENCENDIDO
        set v_on_measured [$Mult_test read_voltage_dc [list range def resolution def]]
        set c_on_measured [$PS_main read_outputs {current}]

        #SE MUESTRA POR PANTALLA LAS CONDICIONES DE ENCENDIDO
        $step add_procedure "Converter switched on at voltage: $v_on_measured   current: $c_on_measured"
    }


    #SE CALCULA LA EFICIENCIA DEL CONVERTIDOR
    if {$v_dl_inpunt >= 0} {
        set efficiency [expr ($v_dl_inpunt*$c_dl_inpunt)/($v_ps_output_mult*$c_ps_output)]
    } else {
        set efficiency 0
    }

    #SE MIDE LA CONDUCTACIA DE LA CARGA DINAMICA Y SE CALCULA LA RESISTENCIA EQUIVALENTE
    set dl_read_resist [expr 1/[DL_test read_conductance  $dl_chan_resist]]

    #SE ESCRIBE EN EL FICHERO UNA LISTA CON TODAS LAS MEDICIONES REALIZADAS EN ESTE PASO.
    puts $fp "$step_counter $v_ps_output_ps $v_ps_output_mult   $c_ps_output    $v_dl_inpunt \
              $c_dl_inpunt  $efficiency     $dl_read_resist"

    #SE INCREMENTA EL CONTADOR DE PASOS COMPLETADOS
    set  step_counter [expr $step_counter+1]

}

#SI DESPUES DE ALCANZAR LA TENSION NOMINAL NO SE DETECTA EL ENCENDIDO DEL CONVERTIDOR, SE REPORTA UN ERROR
if {$c_thres_on == 0} {
    $step rep_error "Converter hasn't been swithched on"
}
```

Código 17. Incremento de la tensión con carga constante

5.3. Tensión constante con carga variable

```
#==============================================================================================
#            T E N S I O N   C O N S T A N T E   C O N   C A R G A   V A R I A B L E
#==============================================================================================

#SE CONFIGURA LA RESISTENCIA INICIAL DE LA CARGA DINAMICA (100 OHMS)
$DL_test set_conductance [list channel $dl_chan_resist conductance [expr 1.0/$dl_resistance_cte] mode auto]

#VARIACION DE LA CARGA DEL CONVERTIDOR {2000 1000 500 100 50}
foreach dl_resistance_val $dl_resistance_secquence {

    #SE CONFIGURA LA RESISTENCIA DE LA CARGA DINAMICA
    $DL_test set_conductance [list channel $dl_chan_resist\
                               conductance [expr 1.0/$dl_resistance_val]\
                               mode auto]

    #ESPERAR TIEMPO DE ESTABILIZACION DE LAS MEDIDAS
    after $tm

    # SE ASIGNAN TODAS LAS MEDICIONES DE CADA (INSTRUMENTO) A SU VARIABLE CORRESPONDIENTE
    # TENSION DE ENTRADA (FUENTE DE ALIMENTACION)
    set v_ps_output_ps [$PS_main read_outputs {voltage}]
    # TENSION DE ENTRADA (MULTIMETRO)
    set v_ps_output_mult [$Mult_test read_voltage_dc [list range $v_ps_output_ps resolution def]]
    # CORRIENTE DE ENTRADA (FUENTE DE ALIMENTACION)
    set c_ps_output [$PS_main read_outputs {current}]
    # TENSION DE SALIDA (CARGA DINAMICA)
    set v_dl_inpunt [$DL_test read_voltage "$dl_chan_resist"]
    # CORRIENTE DE SALIDA (CARGA DINAMICA)
    set c_dl_inpunt [$DL_test read_current "$dl_chan_resist"]

    #SE MUESTRAN LOS RESULTADOS POR PANTALLA
    $step add_procedure "
    Voltage PS output measure (PS measure):          $v_ps_output_ps V
    Current PS output measure (PS measure):          $c_ps_output A
    Voltage dynamic load (Resistance mode):          $v_dl_inpunt V
    Current dynamic load (Resistance mode):          $c_dl_inpunt A
    Voltage PS output measure (multimeter measure): $v_ps_output_mult V
    "

    #SE CALCULA LA EFICIENCIA DEL CONVERTIDOR
    if {$v_dl_inpunt >= 0} {
        set efficiency [expr ($v_dl_inpunt*$c_dl_inpunt)/($v_ps_output_mult*$c_ps_output)]
    } else {
        set efficiency 0
    }

    #SE MIDE LA CONDUCTACIA DE LA CARGA DINAMICA Y SE CALCULA LA RESISTENCIA EQUIVALENTE
    set dl_read_resist [expr 1/[DL_test read_conductance  $dl_chan_resist]]

    #SE ESCRIBE EN EL FICHERO UNA LISTA CON TODAS LAS MEDICIONES REALIZADAS EN ESTE PASO.
    puts $fp "$step_counter $v_ps_output_ps $v_ps_output_mult   $c_ps_output    $v_dl_inpunt \
              $c_dl_inpunt  $efficiency     $dl_read_resist"

    #SE INCREMENTA EL CONTADOR DE PASOS COMPLETADOS
    set  step_counter [expr $step_counter+1]

}

#SE CONFIGURA DE NUEVO LA RESISTENCIA INICIAL DE LA CARGA DINAMICA (100 OHMS)
$DL_test set_conductance [list channel $dl_chan_resist conductance [expr 1.0/$dl_resistance_cte] mode auto]
```

Código 18. Tensión constante con carga variable

5.4. Decremento de la tensión con carga constante

```
#==============================================================================================================
#        D E C R E M E N T O   D E   L A   T E N S I O N   C O N   C A R G A   C O N S T A N T E
#==============================================================================================================
#SE CARGA LA CONFIGURACION DEL OSCILOSCOPIO PARA DETECTAR EL APAGADO DEL CONVERTIDOR (ESCALON DE BAJADA)
$Tek setconfig bajada

#CONDICION DE DECREMENTO DE TENSION: TENSION DE ENTRADA CONVERTIDOR > TENSION MINIMA
while {[$Mult_test read_voltage_dc [list range def resolution def]] > $vmin_down && \
       [$PS_main read_outputs {voltage}] > $vmin_down } {

    #COMPROBAR QUE LA PROTECCION DE LA FUENTE DE ALIMENTACION NO HA REPORTADO NINGUN EVENTO
    set protection_event [$PS_main read_events]

    #SI SE DETECTA UN EVENTO DE PROTECCION. SE PARA LA SECUENCIA Y SE REPORTA UN ERROR.
    if {[expr $protection_event] > 0} {
        end_test
        $step rep_error "Error test stopped due to protection event number: $protection_event"
        break
    }

    #SE GUARDAN LAS MEDICIONES PREVIAS DE TENSION Y CORRIENTE
    set v_prev $v_ps_output_mult
    set c_prev $c_ps_output

    #SE DECREMENTA LA TENSION DE ALIMENTACION.
    set x [expr $x-1]
    $PS_main set_psv [list voltage [expr ($vmin_up+$x*$v_step)]]

    #ESPERAR EL TIEMPO DE ESTABILIZACION DE LAS MEDIDAS
    after $tm

    # SE ASIGNAN TODAS LAS MEDICIONES DE CADA (INSTRUMENTO) A SU VARIABLE CORRESPONDIENTE
    # TENSION DE ENTRADA (FUENTE DE ALIMENTACION)
    set v_ps_output_ps [$PS_main read_outputs {voltage}]
    # TENSION DE ENTRADA (MULTIMETRO)
    set v_ps_output_mult [$Mult_test read_voltage_dc [list range $v_ps_output_ps resolution def]]
    # CORRIENTE DE ENTRADA (FUENTE DE ALIMENTACION)
    set c_ps_output [$PS_main read_outputs {current}]
    # TENSION DE SALIDA (CARGA DINAMICA)
    set v_dl_inpunt [$DL_test read_voltage "$dl_chan_resist"]
    # CORRIENTE DE SALIDA (CARGA DINAMICA)
    set c_dl_inpunt [$DL_test read_current "$dl_chan_resist"]

    #SE MUESTRAN LOS RESULTADOS POR PANTALLA
    $step add_procedure "
    Voltage PS output measure (PS measure):          $v_ps_output_ps  V
    Current PS output measure (PS measure):          $c_ps_output  A
    Voltage dynamic load (Resistance mode):          $v_dl_inpunt  V
    Current dynamic load (Resistance mode):          $c_dl_inpunt  A
    Voltage PS output measure (multimeter measure): $v_ps_output_mult" V

    #COMPROBACION DE ESTADO DE ENCENDIDO DEL CONVERTIDOR: (CORRIENTE_ANTES/CORRIENTE_DESPUES) > 10
    if {[expr abs($c_prev)/abs($c_ps_output)] > $current_peak_detection_value && $c_thres_on == 1} {
        #SE PONE EL FLAG DE APAGADO A 1
        set c_thres_off 1

        #SE EXPORTA LA CAPTURA DEL OSCILOSCOPIO
        $Tek hard_copy bajada

        #SE GUARDA LA TENSION Y CORRIENTE DE APAGADO
        set v_off_measured $v_ps_output_mult
        set c_off_measured $c_ps_output

        #SE MUESTRA POR PANTALLA LAS CONDICIONES DE APAGADO
        $step add_procedure "Converter switched off at voltage: $v_off_measured current: $c_off_measured"
    }
```

```
    #SE CALCULA LA EFICIENCIA DEL CONVERTIDOR
    if {$v_dl_inpunt >= 0} {
        set efficiency [expr ($v_dl_inpunt*$c_dl_inpunt)/($v_ps_output_mult*$c_ps_output)]
    } else {
        set efficiency 0
    }

    #SE MIDE LA CONDUCTACIA DE LA CARGA DINAMICA Y SE CALCULA LA RESISTENCIA EQUIVALENTE
    set dl_read_resist [expr 1/[DL_test read_conductance  $dl_chan_resist]]

    #SE ESCRIBE EN EL FICHERO UNA LISTA CON TODAS LAS MEDICIONES REALIZADAS EN ESTE PASO.
    puts $fp "$step_counter $v_ps_output_ps $v_ps_output_mult   $c_ps_output    $v_dl_inpunt \
              $c_dl_inpunt  $efficiency     $dl_read_resist"

    #SE INCREMENTA EL CONTADOR DE PASOS COMPLETADOS
    set  step_counter [expr $step_counter+1]

}
```

Código 19. Decremento de la tensión con carga constante

5.5. Validación de la prueba

```
#=============================================================================================================
#                            V A L I D A C I O N   D E   L A   P R U E B A
#=============================================================================================================

#SI NO SE HAN DETECTADO AMBOS CAMBIOS DE ESTADO DEL CONVERTIDOR, SE CONSIDERA LA PRUEBA COMO NULA
#SI POR EL CONTRARIO SE HAN DETECTADO AMBOS VALORES, LA PRUEBA SE CONSIDERA VALIDA MOSTRANDO SUS VALORES
if { $c_thres_on == 0 || $c_thres_off == 0 } {
    $step rep_error "Test FAIL"
} else {
    $step add_procedure "Result of the test:
    Hysteresis Voltage on:$v_on_measured    Hysteresis Current on: $c_on_measured
    Hysteresis Voltage off:$v_off_measured Hysteresis Current on: $c_off_measured"
    $step add_procedure "Test SUCCESS"
}

#SE CIERRA EL FICHERO DE MEDIDAS
close $fp

#SE EJECUTA LA SECUENCIA DE FINAL DE TEST
end_test
}
```

Código 20. Validación de la prueba

Capítulo 3

Resultados obtenidos

1. Tabla de mediciones registradas

Los resultados de todas las mediciones se recogerán en la siguiente tabla:

Fase de la secuencia	Medición	Tensión de entrada [V] (F. A)	Tensión de entrada [V] (Multímetro)	Corriente de entrada [A] (F. A.)	Tensión de salida [V] (C.D.)	Corriente de salida [A] (C.D.)	P_{in} [w]	P_{out} [w]	Rendimiento [%]	Carga a la salida del convertidor [Ohms] (C.D.)
1	1	8,00	8,11	0,00	0,00	0,00	0,02	0,00	0,00	100
	2	9,00	9,13	0,00	0,00	0,00	0,02	0,00	0,00	100
1	3	10,01	10,14	0,00	0,00	0,00	0,02	0,00	0,00	100
	4	11,00	11,15	0,00	0,00	0,00	0,02	0,00	0,00	100
Conv. Encendido	5	12,01	12,17	0,22	15,03	0,15	2,66	2,26	0,85	100
	6	13,00	13,18	0,20	15,03	0,15	2,68	2,26	0,84	100
1	7	14,00	14,19	0,19	15,03	0,15	2,67	2,26	0,85	100
	8	15,00	15,21	0,17	15,03	0,15	2,62	2,26	0,86	100
1	9	16,01	16,22	0,16	15,03	0,15	2,67	2,26	0,84	100
	10	17,00	17,23	0,15	15,03	0,15	2,64	2,26	0,86	100
1	11	18,01	18,25	0,15	15,03	0,15	2,65	2,26	0,85	100
	12	18,98	19,25	0,14	15,03	0,15	2,65	2,26	0,85	100
1	13	19,98	20,26	0,13	15,04	0,15	2,63	2,26	0,86	100
	14	20,99	21,28	0,12	15,04	0,15	2,60	2,26	0,87	100
1	15	21,99	22,29	0,12	15,04	0,15	2,63	2,26	0,86	100
	16	22,98	23,30	0,11	15,04	0,15	2,66	2,26	0,85	100
2	17	23,99	24,32	0,11	15,04	0,15	2,68	2,26	0,84	100
	18	23,99	24,32	0,01	15,05	0,01	0,32	0,11	0,35	2000
2	19	23,99	24,32	0,02	15,05	0,02	0,42	0,23	0,54	1000
	20	23,99	24,32	0,03	15,04	0,03	0,70	0,45	0,64	500
2	21	23,99	24,32	0,11	15,04	0,15	2,68	2,26	0,84	100
	22	23,99	24,32	0,22	15,03	0,30	5,23	4,51	0,86	50
2	23	23,99	24,32	0,11	15,04	0,15	2,68	2,26	0,84	100
	24	22,98	23,30	0,11	15,04	0,15	2,66	2,26	0,85	100
3	25	21,99	22,29	0,12	15,04	0,15	2,63	2,26	0,86	100
	26	20,99	21,28	0,12	15,04	0,15	2,60	2,26	0,87	100
3	27	19,98	20,26	0,13	15,04	0,15	2,63	2,26	0,86	100
	28	18,98	19,25	0,14	15,04	0,15	2,65	2,26	0,85	100
3	29	18,01	18,25	0,15	15,04	0,15	2,65	2,26	0,85	100
	30	17,00	17,23	0,15	15,04	0,15	2,64	2,26	0,86	100
3	31	16,00	16,22	0,16	15,04	0,15	2,67	2,26	0,84	100
	32	15,00	15,21	0,17	15,04	0,15	2,62	2,26	0,86	100
3	33	14,00	14,19	0,19	15,04	0,15	2,67	2,26	0,85	100
	34	13,00	13,18	0,20	15,04	0,15	2,63	2,26	0,86	100
3	35	12,01	12,17	0,22	15,04	0,15	2,66	2,26	0,85	100
Conv. Apagado	36	11,00	11,15	0,00	0,07	0,00	0,02	0,00	0,00	100
3	37	10,00	10,14	0,00	0,00	0,00	0,02	0,00	0,00	100
	38	9,00	9,13	0,00	0,00	0,00	0,02	0,00	0,00	100
3	39	8,00	8,11	0,00	0,00	0,00	0,02	0,00	0,00	100
	40	7,00	7,10	0,00	0,00	0,00	0,02	0,00	0,00	100
3	41	6,01	6,09	0,00	0,00	0,00	0,01	0,00	0,00	100
	42	4,99	5,07	0,00	0,00	0,00	0,01	0,00	0,00	100
3	43	4,00	4,06	0,00	0,00	0,00	0,01	0,00	0,00	100

Fase de la secuencia:
1: Incremento de tensión 2: Tensión constante 3: Reducción de tensión

Instrumento que realizó la medida:
F.A.= Fuente alimentación C.D.= Carga dinámica Multímetro

Tabla 4. Tabla de mediciones

2. Resultados mostrados por pantalla

En el siguiente informe se muestra todo el proceso llevado a cabo por el sistema de test, presentado a su vez las mediciones realizadas:

```
==========================================================================================
                   S E Q U E N C E   D E F I N I T I O N
==========================================================================================
SEQUENCE NAME: test_00002_prueba2
DUT: Convertidor_DC_DC
TESTBENCH: hardware_testbench
OBJECTIVE: Verificar especificaciones técnicas del convertidor
==========================================================================================

==========================================================================================
                       S T E P   D E F I N I T I O N
==========================================================================================
STEP ID: 0
OBJECTIVE: Tes0
REQUIREMENTS: none
METHOD: automatic
==========================================================================================


** Note: step 0:    Test settings:
** Note: step 0:
                    Dynamic load resistance:    100  Ohms
                    Minimun voltage rising:     8 V
                    Minimun voltage decreasing: 4 V
                    Maximun voltage:            24 V
                    Voltage step:               1 V
                    Current on:                 0.005 A
                    Current off:                0.010 A
                    Step time (setting time):   1000  ms

** Note: step 0:    Check that all devices connected
** Note: step 0:    setup DL as resistance: 100
** Note: step 0:
                    ######################################################################
                    #                        Increasing voltage test:
                    ######################################################################
** Note: step 0:    Voltage PS output measure (PS measure):         7.99760E+0  V
                    Current PS output measure (PS measure):         -2.19186E-3  A
                    Voltage dynamic load (Resistance mode):         0.002  V
                    Current dynamic load (Resistance mode):         0.00008  A
                    Voltage PS output measure (multimeter measure): +8.10963880E+00  V

** Note: step 0:                            ***
                    Voltage PS output measure (PS measure):         1.20069E+1  V
                    Current PS output measure (PS measure):         2.19053E-1  A
                    Voltage dynamic load (Resistance mode):         15.031  V
                    Current dynamic load (Resistance mode):         0.15007  A
                    Voltage PS output measure (multimeter measure): +1.21652720E+01  V
** Note: step 0:    ----------------------------------------------------------------------
                    Converter switched on at voltage: +1.21653260E+01 V  current: 2.19053E-1 A
                    ----------------------------------------------------------------------
** Note: step 0:
                    Voltage PS output measure (PS measure):         1.30020E+1  V
                    Current PS output measure (PS measure):         2.03527E-1  A
                    Voltage dynamic load (Resistance mode):         15.032  V
                    Current dynamic load (Resistance mode):         0.15007  A
                    Voltage PS output measure (multimeter measure): +1.31762440E+01  V

                                            ***
** Note: step 0:
                    Voltage PS output measure (PS measure):         2.39869E+1  V
                    Current PS output measure (PS measure):         1.10371E-1  A
                    Voltage dynamic load (Resistance mode):         15.035  V
                    Current dynamic load (Resistance mode):         0.15009  A
                    Voltage PS output measure (multimeter measure): +2.43188170E+01  V
** Note: step 0:
** Note: step 0:    ######################################################################
** Note: step 0:    #                        Constant voltage:
** Note: step 0:    ######################################################################
** Note: step 0:
                    Voltage PS output measure (PS measure):         2.39869E+1  V
                    Current PS output measure (PS measure):         1.33341E-2  A
                    Voltage dynamic load (Resistance mode):         15.046  V
                    Current dynamic load (Resistance mode):         0.00755  A
                    Voltage PS output measure (multimeter measure): +2.43218140E+01  V

                                            ***
** Note: step 0:
                    Voltage PS output measure (PS measure):         2.39869E+1  V
                    Current PS output measure (PS measure):         2.15171E-1  A
                    Voltage dynamic load (Resistance mode):         15.028  V
                    Current dynamic load (Resistance mode):         0.29994  A
                    Voltage PS output measure (multimeter measure): +2.43155640E+01  V
** Note: step 0:
** Note: step 0:    ######################################################################
** Note: step 0:    #                    Decreasing voltage test:
** Note: step 0:    ######################################################################
** Note: step 0:
** Note: step 0:
                    Voltage PS output measure (PS measure):         2.39869E+1  V
                    Current PS output measure (PS measure):         1.10371E-1  A
                    Voltage dynamic load (Resistance mode):         15.036  V
                    Current dynamic load (Resistance mode):         0.15010  A
                    Voltage PS output measure (multimeter measure): +2.43188700E+01  V

                                            ***
** Note: step 0:
                    Voltage PS output measure (PS measure):         1.10022E+1  V
                    Current PS output measure (PS measure):         -2.19186E-3  A
                    Voltage dynamic load (Resistance mode):         0.066  V
                    Current dynamic load (Resistance mode):         0.00009  A
                    Voltage PS output measure (multimeter measure): +1.11525880E+01  V
** Note: step 0:    ----------------------------------------------------------------------
                    Converter switched off at voltage: +1.11525880E+01 V   current:-2.19186E-3 A
                    ----------------------------------------------------------------------
** Note: step 0:
                    Voltage PS output measure (PS measure):         9.99746E+0  V
                    Current PS output measure (PS measure):         -2.19186E-3  A
                    Voltage dynamic load (Resistance mode):         0.003  V
                    Current dynamic load (Resistance mode):         0.00008  A
                    Voltage PS output measure (multimeter measure): +1.01415820E+01  V

                                            ***
** Note: step 0:
                    Voltage PS output measure (PS measure):         3.99789E+0  V
                    Current PS output measure (PS measure):         -2.19186E-3  A
                    Voltage dynamic load (Resistance mode):         0.002  V
                    Current dynamic load (Resistance mode):         0.00008  A
                    Voltage PS output measure (multimeter measure): +4.05764040E+00  V
** Note: step 0:
                    ######################################################################
                    #                    Result of the test:
                    ######################################################################
** Note: step 0:    ----------------------------------------------------------------------
                    Hysteresis Voltage on:+1.21653260E+01 V  Hysteresis Current on: 2.19053E-1 A
                    Hysteresis Voltage off:+1.11525880E+01 V  Hysteresis Current on:-2.19186E-3 A
                    ----------------------------------------------------------------------
** Note: step 0:    Test: SUCCESS
```

Código 21. Validación de la prueba.

3. Capturas del osciloscopio

Con el fin de poder observar de forma más visual las distintas transiciones del convertidor entre sus distintos estados, se programa la secuencia para que capture las dos siguientes formas de onda desde el osciloscopio:

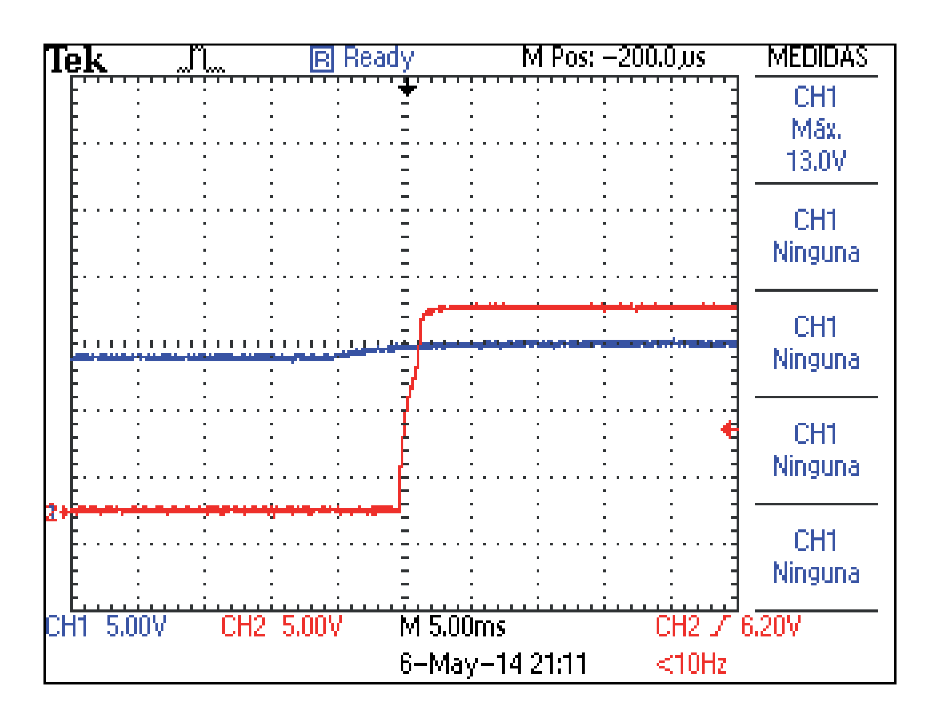


Figura 22. Captura del instante en que se enciende el convertidor.

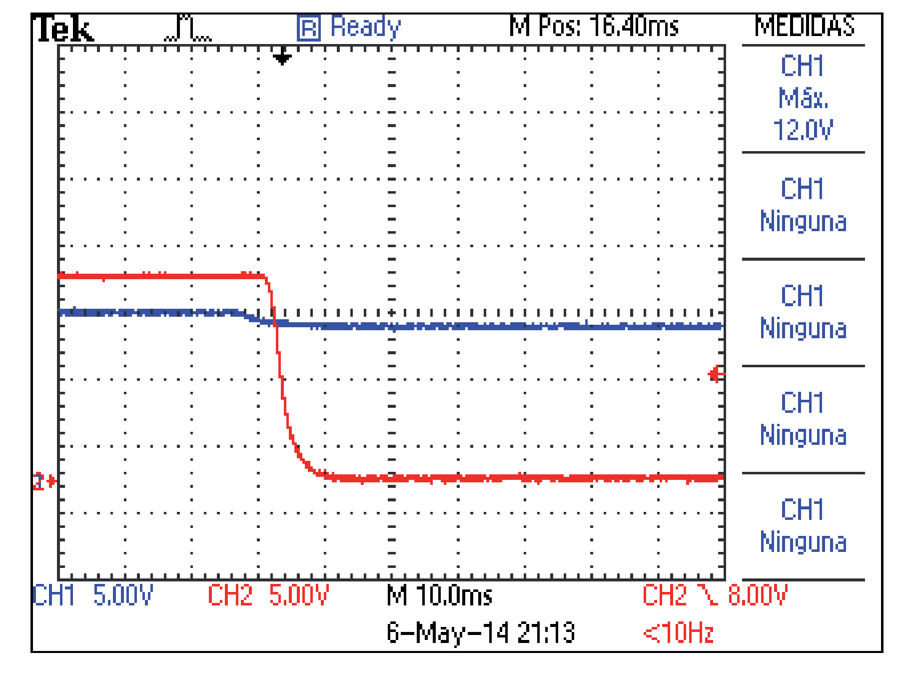


Figura 23. Captura del instante en que se apaga el convertidor.

4. Análisis y estudio de los resultados

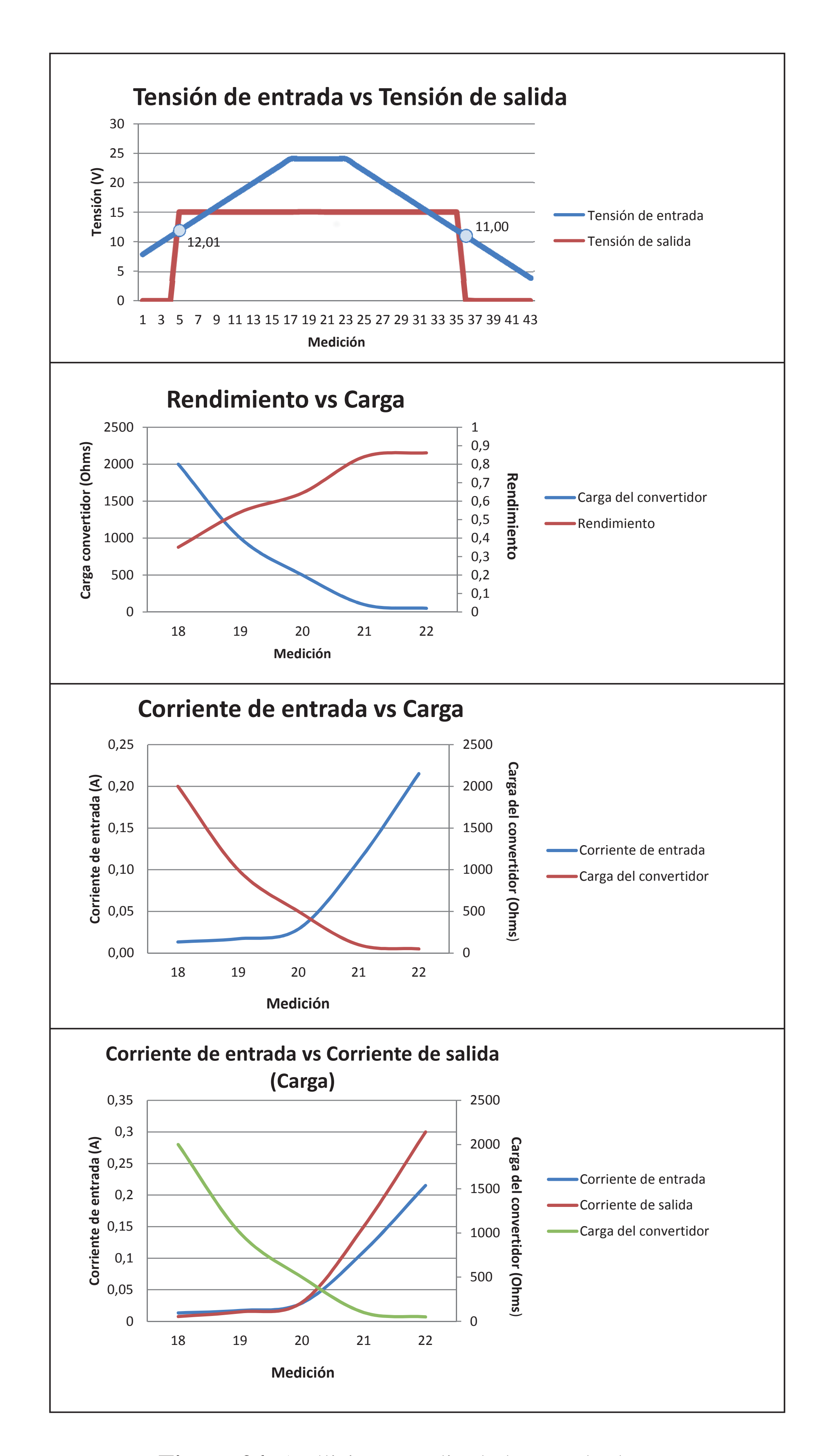


Figura 24. Análisis y estudio de los resultados.

5. Resultados de la verificación

	V_{on} [V]	V_{off} [V]	V_{out} [V]	η_{med}
Espec. Técnicas	12	11	15	84%
Medición	12,17	11,15	15,035	85,25%
Cumple con las especificaciones	✓	✓	✓	✓

Figura 25. Resultados de la prueba.

Se puede observar cómo, además de verificar las especificaciones técnicas facilitadas por el fabricante, se han podido realizar otros muchos estudios sobre el comportamiento del convertidor simulando distintas condiciones de trabajo a distintos niveles de carga. Tales estudios pueden ser de utilidad a la hora de implementar el convertidor en una tarjeta de potencia u otro dispositivo electrónico.

La ejecución de secuencias de test totalmente automatizadas permite verificar de forma rápida y eficiente cualquier dispositivo electrónico; cuya verificación, en el caso de ser llevada a cabo mediante un sistema de test manual o semiautomático, podría llevar varias horas de trabajo o incluso días.

PARTE V

CONCLUSIONES Y FUTUROS DESARROLLOS



Metodologías que hacen posibles los proyectos

El resultado final de todo proyecto, queda intrínsecamente configurado por el tipo de metodología implantada en dicho proyecto.

Muchos serán los que piensen en un primer momento, que una metodología aplicada al diseño de un sistema de test debe estar estar centrada solo y exclusivamente al mero diseño del sistema. Pero la realidad va más allá. La metodología aplicada en este proyecto no solamente se ha centrado en el diseño de un sistema de test, sino que también ha permitido el poder llevarlo a cabo cumpliendo con las altas expectativas sobre la escalabilidad y flexibilidad del sistema. Todo ello gracias al haber aplicado en cada una de las partes del proyecto una misma metodología previamente establecida y aceptada por todos los componentes del grupo.

Una metodología bien cimentada permite establecer un seguimiento en detalle de todos los aspectos del proyecto que son clave en la ejecución del mismo: La planificación de tiempos, la arquitectura del software, la gestión de versiones de un programa en desarrollo... son algunos de los aspectos relacionados directamente con el proceso de elaboración de un proyecto, y que deben estar recogidos en su metodología.

Una de las riquezas mejor guardadas dentro de la filosofía de trabajo de una empresa es su metodología. Gracias a ella muchos proyectos proyectos salen adelante, pero poca gente es capaz de ver que desde sus inicios hasta la entrega final del producto, todo el proyecto es pura metodología.

Menos visual, más eficiente

Uno de los elementos a los que se ha tenido que renunciar al pasar de un sistema de test semiautomático a uno totalmente automatizado, es el aspecto visual del entorno de trabajo. (De momento)

Las secuencias de test llevadas a cabo desde un sistema de test manual, permitían de forma muy gráfica, el poder controlar los instrumentos de forma remota, visualizando con colores y formas el estado de la secuencia.

En este proyecto, por el contrario, se ha dado prioridad a la eficiencia, por lo que el aspecto visual se ha limitado a breves lineas de texto donde aparecen reflejadas todas las mediciones realizadas.

Una interfaz gráfica no es, ni mucho menos, incompatible con los sistemas de test automatizados. Pero la elaboración de toda una interfaz gráfica que sea realmente eficiente y contenga más funciones que las de simplemente mostrar por pantalla el estado de la secuencia , hace que esta tarea se convierta en un nuevo proyecto de expansión al sistema de test actual.

El verdadero potencial de la automatización

Como se ha podido apreciar en los resultados de la secuencia anteriormente mostrados, no solamente se ha podido comprobar que el componente cumple con las especificaciones del fabricante, sino que se han podido llevar a cabo otros muchos estudios adicionales sobre el comportamiento del mismo. Esto solo es posible gracias a la gran cantidad de muestras recogida por el sistema.

Esto solo es posible gracias a la implantación de una lógica programable en la estructura de cada una de las secuencias. Los sistemas de test semiautomáticos, en la mayoría de los casos, solamente pueden mandar al instrumento lineas de comandos de forma secuencial, imposibilitando la utilización de bucles. Esto hace que el tiempo de ejecución de cada secuencia sea notoriamente superior al requerido por un sistema de test automatizado.

En proyectos donde el número de mediciones sea tan elevado que sobrepase la capacidad estándar de los instrumentos de medida, se utilizarán matrices de puntos de cruce como módulos de conmutación. De esta forma se conseguirá ampliar el número de mediciones posibles hasta satisfacer las necesidades del cliente.



Figura 26. Ejemplo de módulo de conmutación por matriz de puntos de cruce

Futuros desarrollos

Una de las grandes ventajas de este sistema de test, como ya se ha indicado en numerosas ocasiones, es su gran escalabilidad. Esto permite que futuros proyectos puedan seguir desarrollando nuevas funcionalidades que puedan ser añadidas al sistema actual. Entre ellas cabría destacar:

1. Comandado de instrumentos por un sistema de control avanzado (Tarjeta FPGA)

El gran potencial de procesado presente en dichas tarjetas, hace que sean el compañero de viaje perfecto para el comandado de instrumentos en sistemas de alto desempeño.

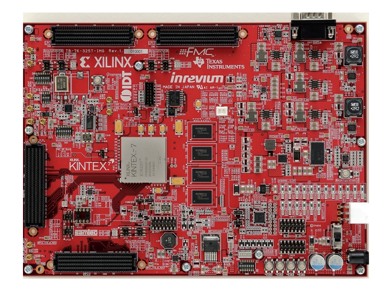


Figura 27. Ejemplo de FPGA

2. Diseño de un interfaz gráfico para el sistema de test

Hacer que el sistema de test sea más intuitivo y fácil de usar, desarrollando una interfaz gráfica que permita tanto mostrar el estado de la secuencia, como la configuración de la misma.

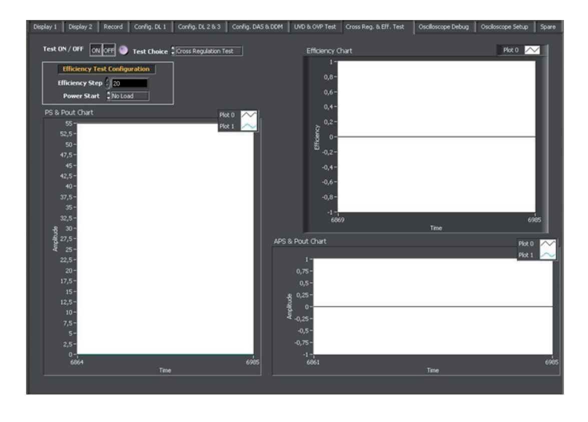


Figura 28. Ejemplo de Interfaz gráfica del sistema de test

3. Ampliar el número de dispositivos controlados por el sistema de test

En el caso de que fuese necesario la utilización de nuevos dispositivos en la secuencia de test, podrían implementarse nuevas librerías de comandado específicas para cada instrumento.



Bibliografía

[1] METODOLOGÍA DE PRUEBAS (TESTING) *Globons IT Solutions*
http://www.globons.com/metodologia-testing.php
Último acceso: Mayo 2014.

[2] METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE (WIKI DE LA ASIGNATURA)
Universidad de oriente - Maturín, Mónagas, Venezuela
http://wiki.monagas.udo.edu.ve/index.php/Metodologías_para_el_desarrollo_de_software
Último acceso: Mayo 2014.

[3] APTEST - SOFTWARE TESTING GLOSSARY
http://www.aptest.com/glossary.html
Último acceso: Mayo 2014.

[4] IAN KUON, AARON EGIER y JONATHAN ROSE, *Design, Layout and Verification of an FPGA using Automated Tools* , University of Toronto, Toronto, Ontario, Canada 2004.

[5] STALLINGS, W., *Comunicaciones y redes de computadores*, Ed. Prentice-Hall. 2007.

[6] DEP. ELECTRÓNICA, *Universidad Regional de Buenos Aires*

[7] M.C. CARLOS MARTÍNEZ, *National Instruments* ® - *Comparación de Buses de Instrumentos para Pruebas Automatizadas* , 2007

DOCUMENTO II

MANUAL DE USUARIO



Índice de librerías

Capítulo 1

Gestor de comunicaciones: driver_gpib

Librería destinada a la gestión de comunicaciones de todos los instrumentos, conectados por GPIB, que compongan el sistema de test.

Instanciación:

Parámetros del constructor:

log_name: Lista con el directorio hasta el archivo de registro de comandos y el nombre del archivo.

list_devices: Lista con los instrumentos que van a participar dentro del sistema de test

Ejemplo de instanciación:

```
set d_gpib [driver_gpib d_gpib $driver_log_name $inlist]
```

Funciones de la clase:

1. Escaneo de dispositivos conectados.

Función:

get_devices_connected

Parámetros de entrada:

verbose? (True | False)

Parámetros de salida / acciones:

verbose: True (devuelve la información en variables distintas):
"GPIB0::2::INSTR gpib HEWLETT-PACKARD,34401A"
"GPIB0::5::INSTR gpib HEWLETT-PACKARD,6653A"

verbose: False (devuelve la información como una variable en formato de lista)
{GPIB0::2::INSTR gpib HEWLETT-PACKARD,34401A}{GPIB0::5::INSTR gpib HEWLETT-PACKARD,6653A}

2. Comprobar conexión con todos los instrumentos pertenecientes a la prueba de test.

Función:

check_devices_connected

Parámetros de entrada:

Ninguno

Parámetros de salida / acciones:

1: Todos los dispositivos están conectados
0: Alguno de los dispositivos no están conectados

3. Registro de todos los comandos enviados por el bus GPIB.

Función:

command_sent_wfile

Parámetros de entrada:

dev_name: Nombre del instrumento

dev_addrs: Dirección GPIB del instrumento

command: Comando enviado al instrumento

Parámetros de salida / acciones:

Archivo con la siguiente información:

04/30/2014 17:14:56 GPIB0::5::INSTR "MEAS:VOLT?"
04/30/2014 17:14:56 GPIB0::5::INSTR "MEAS:CURR?"
04/30/2014 17:16:05 GPIB0::1::INSTR "INST:COUP CH1,"

Capítulo 2

Gestor de errores: error_handle

Librería destinada a la gestión de errores producidos durante el comandado del instrumento. Las funciones principales de esta librería se basarán en un apagado controlado en situaciones de emergencia y en mostrar los errores por pantalla.

Instanciación:

Parámetros del constructor:

Ninguno

Ejemplo de instanciación:

```
set e_h [error_handle e_h]
```

Funciones de la clase:

1. Reportar los errores producidos por pantalla.

Función:

error_report_screen

Parámetros de entrada:

dev_addrs: Dirección GPIB del instrumento.

error_data: Lista con el código de error y su descripción.

Parámetros de salida / acciones:

Muestra por pantalla la siguiente información:

```
GPIB0::8::INSTR "-101" "Invalid character"
GPIB0::8::INSTR "-102" "Syntax error"
```

2. Añadir referencia del instrumento para poder comandarlo.

Función:

add_instrument

Parámetros de entrada:

instrument_id: Nombre del objeto asociado al instrumento.

Ej: escaner_test

instr_ref: Referencia al objeto

Ej: ::escaner_test::

Parámetros de salida / acciones:

Añade a una lista los parámetros recibidos junto a los del resto de instrumentos. Dicha lista será instanciada en el momento en el que se vaya a ejecutar el apagado controlado.

3. Cargar secuencia de parada controlada.

Función:

load_controlled_exit

Parámetros de entrada:

controlled_exit: Lista que contiene cada uno de los pasos de la secuencia que será ejecutada en situación de emergencia.

Parámetros de salida / acciones:

Almacena en una variable la lista de comandos.

4. Gestor de situaciones de emergencia. Parada controlada.

Función:

execute_controlled_exit

Parámetros de entrada:

severity: Nivel de severidad reportado por el instrumento

Parámetros de salida / acciones:

Dependiendo del nivel de severidad del error producido, se decidirá ejecutar la secuencia de apagado controlado previamente establecida o simplemente reportarlo como error leve.

En circuitos simples es común que lo primero que sea desconectado sea la fuente de alimentación seguido de las cargas dinámicas.

Capítulo 3

Fuente de alimentación: driver_gpib_hp6653a

Librería destinada al comandado de la fuente de alimentación HP 6653A.

Instanciación:

Parámetros del constructor:

visa_addrs: Dirección GPIB asignada al instrumento

d_gpib: Referencia al objeto de la clase driver_gpib

d_error_handle: Referencia al objeto de la clase error_handle

Ejemplo de instanciación:

```
set PS_main [driver_gpib_hp6653a PS_main "GPIB0::5::INSTR" $d_gpib $e_h]
```

Funciones de la clase:

1. Encender | Apagar la pantalla del instrumento.

Función:

display

Parámetros de entrada:

on_off: (on | off) Selecciona el estado de la pantalla

Parámetros de salida / acciones:

Enciende o apaga la pantalla del instrumento

2. Activar | Desactivar la alimentación.

Función:

output

Parámetros de entrada:

output_conf: (on | off) Habilita o deshabilita la alimentación.

Parámetros de salida / acciones:

Enciende o apaga la alimentación de la fuente.

3. Configurar tensión y corriente.

Función:

set_ps_vc

Parámetros de entrada:

output_vc: Lista con la configuración de tensión y corriente:

Ejemplo: [list voltage {50 mV} current {3 mA}]

- Voltage: nivel de tension {<value> V| mV...MIN|MAX}
- Current: nivel de corriente {<value> A| mA...MIN|MAX}

Parámetros de salida / acciones:

Configura la tensión y corriente de salida sin cambiar el estado de la alimentación

4. Configurar tensión.

Función:

set_psv

Parámetros de entrada:

output_v: Lista con la configuración de tensión:

Ejemplo: [list voltage {50 mV}]

- Voltage: nivel de tensión {<value> V| mV...MIN|MAX}

Parámetros de salida / acciones:

Configura la tensión de salida sin cambiar el estado de la alimentación

5. Configurar corriente.

Función:

set_ps_c

Parámetros de entrada:

output_c: Lista con la configuración de corriente:

Ejemplo: [list current {3 mA}]

- Current: nivel de corriente {<value> A| mA...MIN|MAX}

Parámetros de salida / acciones:

Configura la corriente de salida sin cambiar el estado de la alimentación

6. Configurar protecciones de tensión y corriente

Función:

set_prot_psvc

Parámetros de entrada:

protect_vc: Lista con la configuración de protección de tensión y corriente:

Ejemplo: [list v_level {1 V} c_level {60 mA}]

- **v_level:** nivel de tensión (protección) {<value> V | mV...MIN|MAX}
- **c_level:** nivel de corriente (protección) {<value> A | mA...MIN|MAX}

Parámetros de salida / acciones:

Configura la protección de tensión y de corriente acorde a los valores indicados después de v_level (tensión) y c_level (corriente).

7. Configurar protecciones de tensión.

Función:

set_prot_psv

Parámetros de entrada:

protect_v: Lista con la configuración de protección de tensión:

Ejemplo: [list overv_level {3 V}]

- **overv_level:** nivel de tensión (protección) {<value> V | mV...MIN|MAX}

Parámetros de salida / acciones:

Configura la protección de tensión acorde al valor indicado después de overv_level. Esta función no cambia la configuración de la tensión de salida.

8. Configurar protecciones de corriente.

Función:

set_prot_psc

Parámetros de entrada:

protect_c: Lista con la configuración de protección de corriente:

Ejemplo: [list c_level {3 A}]

- **c_level:** nivel de corriente (protección) {<value> A | mA...MIN|MAX}

Parámetros de salida / acciones:

Configura la protección de corriente acorde al valor indicado después de c_level. Esta función no cambia la configuración de la corriente de salida.

9. Eliminar configuración de protecciones.

Función:

clear_prot

Parámetros de entrada:

Ninguno

Parámetros de salida / acciones:

Elimina la protección configurada y cualquier flag de protección activo.
Un flag de protección se activa cuando se sobrepasa cualquiera de los valores de protección previamente configurados.

10. Leer flags (eventos) de protección.

Función:

read_events

Parámetros de entrada:

Ninguno

Parámetros de salida / acciones:

Devuelve, en formato numérico, cualquier evento de protección ocurrido:
OV (Over Voltage): 1
OC (Over Current): 2
OT (Over Temperature): 16
RI (Remote inhibit is active): 512
UNR (Power supply output is unregulated): 1024

11. Leer valores de salida [Tensión | Corriente | Potencia].

Función:

read_outputs

Parámetros de entrada:

type: parámetro con el nombre de la medida {voltage | current | power}

Parámetros de salida / acciones:

Devuelve la medida indicada en sus unidades correspondientes:

- Voltaje (V)
- Corriente (A)
- Potencia (W)

Capítulo 4

Multímetro: driver_gpib_hp34401a

Librería destinada al comandado del multímetro HP 34401A.

Instanciación:

Parámetros del constructor:

visa_addrs: Dirección GPIB asignada al instrumento

d_gpib: Referencia al objeto de la clase driver_gpib

d_error_handle: Referencia al objeto de la clase error_handle

Ejemplo de instanciación:

```
set Mult_test [driver_gpib_hp34401a Mult_test "GPIB0::2::INSTR" $d_gpib $e_h]
```

Funciones de la clase:

1. Encender | Apagar la pantalla del instrumento.

Función:

display

Parámetros de entrada:

on_off: (on | off) Selecciona el estado de la pantalla

Parámetros de salida / acciones:

Enciende o apaga la pantalla del instrumento

2. Medir resistencia de dos cables.

Función:

read_resx2

Parámetros de entrada:

r_confx2: Lista con la configuración del multímetro para medir resistencias de dos cables.

Ejemplo: [list range {5 ohm} resolution {1 ohm}]

- range: rango de medición
  { <value> MOHM | KOHM | OHM ... | MIN | MAX | DEF }
- resolution: resolución de la medida
  { <value> MOHM | KOHM | OHM ... | MIN | MAX | DEF }

Parámetros de salida / acciones:

Devuelve el valor de la resistencia medida para un rango y una resolución determinados.

Unidades de la medida: Ohms

3. Medir resistencia de cuatro cables.

Función:

read_resx4

Parámetros de entrada:

r_confx4: Lista con la configuración del multímetro para medir resistencias de cuatro cables.

Ejemplo: [list range {5 ohm} resolution {1 ohm}]

- range: rango de medición
  { <value> MOHM | KOHM | OHM ... | MIN | MAX | DEF }
- resolution: resolución de la medida
  { <value> MOHM | KOHM | OHM ... | MIN | MAX | DEF }

Parámetros de salida / acciones:

Devuelve el valor de la resistencia medida para un rango y una resolución determinados.

Unidades de la medida: Ohms

4. Medir tensión DC.

Función:

read_voltage_dc

Parámetros de entrada:

v_conf_dc: Lista con la configuración del multímetro para medir una tensión DC.

Ejemplo: [list range {5 mV} resolution {1 mV}]

- range: rango de medición { <value> kV | V | mV... | MIN | MAX | DEF }
- resolution: resolución de la medida {<value> kV | V | mV... | MIN | MAX | DEF }

Parámetros de salida / acciones:

Devuelve el valor de la tensión DC medida para un rango y una resolución determinados.

Unidades de la medida: V

5. Medir tensión AC.

Función:

read_voltage_ac

Parámetros de entrada:

v_conf_ac: Lista con la configuración del multímetro para medir una tensión AC.

Ejemplo: [list range {5 mV} resolution {1 mV}]

- range: rango de medición { <value> kV | V | mV... | MIN | MAX | DEF }
- resolution: resolución de la medida {<value> kV | V | mV... | MIN | MAX | DEF }

Parámetros de salida / acciones:

Devuelve el valor de la tensión AC medida para un rango y una resolución determinados.

Unidades de la medida: V

6. Medir corriente DC.

Función:

read_current_dc

Parámetros de entrada:

c_conf_dc: Lista con la configuración del multímetro para medir una corriente DC.

Ejemplo: [list range {20 A} resolution {1 mA}]

- range: rango de medición { <value> A | mA ... | MIN | MAX | DEF}
- resolution: resolución de la medida { <value> A | mA ... | MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve el valor de la corriente DC medida para un rango y una resolución determinados.

Unidades de la medida: A

7. Medir corriente AC.

Función:

read_current_ac

Parámetros de entrada:

c_conf_ac: Lista con la configuración del multímetro para medir una corriente AC.

Ejemplo: [list range {20 A} resolution {1 mA}]

- range: rango de medición { <value> A | mA ... | MIN | MAX | DEF}
- resolution: resolución de la medida { <value> A | mA ... | MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve el valor de la corriente AC medida para un rango y una resolución determinados.

Unidades de la medida: A

8. Medir frecuencia.

Función:

read_freq

Parámetros de entrada:

f_conf: Lista con la configuración del multímetro para medir frecuencia.

Ejemplo: [list range {20 kHz} resolution {100 hz}]

- range: rango de medición { <value> kHz | Hz ... | MIN | MAX | DEF}
- resolution: resolución de la medida { <value> kHz | Hz ... | MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve el valor de la frecuencia medida para un rango y una resolución determinados.

Unidades de la medida: Hz

9. Repetir última medición.

Función:

read_meas

Parámetros de entrada:

Ninguno

Parámetros de salida / acciones:

Devuelve el valor correspondiente a la última medición realizada. La configuración del multímetro (rango y resolución) se mantienen constantes.

Unidades de la medida: Depende de la última medición realizada

Capítulo 5

Scanner: driver_gpib_agil34970a

Librería destinada al comandado del escáner Agilent 34970A.

Instanciación:

Parámetros del constructor:

visa_addrs: Dirección GPIB asignada al instrumento

d_gpib: Referencia al objeto de la clase driver_gpib

d_error_handle: Referencia al objeto de la clase error_handle

Ejemplo de instanciación:

```
set Scn_test [driver_gpib_agil34970a Scn_test "GPIB0::8::INSTR" $d_gpib $e_h]
```

Funciones de la clase:

1. Encender | Apagar la pantalla del instrumento.

Función:

display

Parámetros de entrada:

on_off: (on | off) Selecciona el estado de la pantalla

Parámetros de salida / acciones:

Enciende o apaga la pantalla del instrumento

2. Medir tensión DC.

Función:

read_voltage_dc

Parámetros de entrada:

v_conf_dc: Lista con la configuración del escáner para medir tensión DC.

Ejemplo: [list channel {105:110,215} range {5 mV} resolution {1 mV}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  { (:) (From : to) | (,) single }
- range: rango de medición {<value> V | mV ... | MIN | MAX | DEF}
- resolution: resolución de la medida {<value> V | mV ... | MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve una lista con todas las mediciones realizadas.

Unidades de la medida: V

3. Medir tensión AC.

Función:

read_voltage_ac

Parámetros de entrada:

v_conf_ac: Lista con la configuración del escáner para medir tensión AC.

Ejemplo: [list channel {105:110,215} range {5 mV} resolution {1 mV}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  { (:) (From : to) | (,) single }
- range: rango de medición {<value> V | mV ... | MIN | MAX | DEF}
- resolution: resolución de la medida {<value> V | mV ... | MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve una lista con todas las mediciones realizadas.

Unidades de la medida: V

4. Medir corriente DC.

Función:

read_current_dc

Parámetros de entrada:

c_conf_dc: Lista con la configuración del escáner para medir corriente DC.

Ejemplo: [list channel {105:110,215} range {200 mA} resolution {1 mA}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  { (:) (From : to) | (,) single }
- range: rango de medición {<value> A | mA ... | MIN | MAX | DEF }
- resolution: resolución de la medida {<value> A | mA ... | MIN | MAX | DEF }

Parámetros de salida / acciones:

Devuelve una lista con todas las mediciones realizadas.

Unidades de la medida: A

5. Medir corriente AC.

Función:

read_current_ac

Parámetros de entrada:

c_conf_ac: Lista con la configuración del escáner para medir corriente AC.

Ejemplo: [list channel {105:110,215} range {200 mA} resolution {1 mA}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  { (:) (From : to) | (,) single }
- range: rango de medición {<value> A | mA ... | MIN | MAX | DEF }
- resolution: resolución de la medida {<value> A | mA ... | MIN | MAX | DEF }

Parámetros de salida / acciones:

Devuelve una lista con todas las mediciones realizadas.

Unidades de la medida: A

6. Medir resistencia de dos cables.

Función:

read_resistancex2

Parámetros de entrada:

res_conf: Lista con la configuración del escáner para medir resistencias de dos cables.

Ejemplo: [list channel {105:110,215} range {2 ohm} resolution {1 mohm}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  { (:) (From : to) | (,) single }
- range: rango de medición {<value> ohm | kohm ...| MIN | MAX | DEF}
- resolution: resolución de la medida {<value> ohm | kohm ...| MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve una lista con todas las mediciones realizadas.

Unidades de la medida: Ohms.

7. Medir resistencia de cuatro cables.

Función:

read_resistancex4

Parámetros de entrada:

res_conf: Lista con la configuración del escáner para medir resistencias de cuatro cables.

Ejemplo: [list channel {105:110,215} range {2 ohm} resolution {1 mohm}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  { (:) (From : to) | (,) single }
- range: rango de medición {<value> ohm | kohm ...| MIN | MAX | DEF}
- resolution: resolución de la medida {<value> ohm | kohm ...| MIN | MAX | DEF}

Parámetros de salida / acciones:

Devuelve una lista con todas las mediciones realizadas.

Unidades de la medida: Ohms

Capítulo 6

Carga dinámica: driver_gpib_kikuplz150u

Librería destinada al comandado de la carga dinámica Kikusui PLZ150u.

Instanciación:

Parámetros del constructor:

visa_addrs: Dirección GPIB asignada al instrumento

d_gpib: Referencia al objeto de la clase driver_gpib

d_error_handle: Referencia al objeto de la clase error_handle

Ejemplo de instanciación:

```
set DL_test[driver_gpib_kikuplz150u DL_test "GPIB0::1::INSTR" $d_gpib $e_h]"
```

Funciones de la clase:

1. Elección de modo de trabajo.

Función:

set_mode

Parámetros de entrada:

l_dl_mode: Lista con la configuración del modo de trabajo de la carga dinámica.

Ejemplo: [list channel {ch1 ch4} mode {cc}]

- channel: Indica el canal/es por los que se va a realizar la medida.
  {CH1, CH2, CH3... ALL | NONE}
- mode: modo de trabajo {CC | CR |CV | CCCV | CRCV }

Modos posibles de trabajo:
CC: Constant current
CR: Constant resistance mode
CV: Constant voltage mode
CCCV: Constant current mode + constant voltage mode

Parámetros de salida / acciones:

Configura el modo de trabajo de los canales especificados.

2. Activar | Desactivar el consumo de potencia.

Función:

input

Parámetros de entrada:

state: (on | off) Selecciona el estado del consumo de potencia.

Parámetros de salida / acciones:

Habilita o deshabilita el consumo de potencia por parte de la carga dinámica.

3. Activar | Desactivar la fuente de alimentación.

Función:

output

Parámetros de entrada:

state: (on | off) Selecciona el estado de la salida de potencia.

Parámetros de salida / acciones:

Habilita o deshabilita la generación de potencia por parte de la carga dinámica.

4. Configurar protección [Corriente | Tensión | Potencia].

Función:

set_protection

Parámetros de entrada:

l_dl_protection: Lista con la configuración de las protecciones de corriente, tensión y potencia de la carga dinámica

Ejemplo: [list channel {ch1 ch2} current {1 A} voltage {no} power {MAX}]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- current: nivel de corriente permitido {<value> mA | A... MIN | MAX | no}
- voltage: nivel de tensión permitido {<value> mV | V... MIN | MAX | no}
- power: nivel de potencia permitido {<value> W... MIN | MAX | no}

Interpretación de valores:

- MAX: Se considera "deshabilitar" la protección.
- No: La configuración anterior no será modificada.

Parámetros de salida / acciones:

Habilita, o no modifica, la configuración de cada una de las protecciones de la carga dinámica

5. Deshabilitar protección y reiniciar flags de protección.

Función:

disable_prot

Parámetros de entrada:

Ninguno

Parámetros de salida / acciones:

Deshabilita todas las protecciones y flags de protección de la carga dinámica.

6. Configurar conductancia.

Función:

set_conductance

Parámetros de entrada:

l_dl_conductance: Lista con la configuración de carga resistiva.

Ejemplo: [list channel {ch1 ch2} conductance {0.2 sie} mode auto]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- conductance: nivel de conductancia {<value> sie | msie}
- mode: módulo de conductancia {AUTO | LOW | MED | HIGH}

Información adicional:

- El parámetro mode permite escoger el nivel de carga utilizado. Un módulo alto permite utilizar valores de módulos inferiores, pero no al revés.
- Si no se conoce los márgenes de trabajo de cada nivel, se recomienda utilizar el modo AUTO.

Parámetros de salida / acciones:

Configura la carga dinámica como una carga resistiva.

7. Configurar corriente de entrada.

Función:

set_current

Parámetros de entrada:

l_dl_current: Lista con la configuración de corriente contante.

Ejemplo: [list channel {ch1 ch2} current {1.2 A} mode MED]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- current: nivel de corriente {value mA | A... MIN | MAX}
- mode: módulo de corriente {AUTO | LOW | MED | HIGH}

Información adicional:

- El parámetro mode permite escoger el nivel de carga utilizado. Un módulo alto permite utilizar valores de módulos inferiores, pero no al revés.
- Si no se conoce los márgenes de trabajo de cada nivel, se recomienda utilizar el modo AUTO.

Parámetros de salida / acciones:

Configura la carga dinámica en modo de corriente constante.

8. Configurar tensión de salida.

Función:

set_voltage

Parámetros de entrada:

l_dl_voltage: Lista con la configuración de tensión contante.

Ejemplo: [list channel {ch1} voltage {5 V} mode auto]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- voltage: nivel de tensión {value mV | V... MIN | MAX}
- mode: módulo de tensión {AUTO | LOW | MED | HIGH}

Información adicional:

- El parámetro mode permite escoger el nivel de carga utilizado. Un módulo alto permite utilizar valores de módulos inferiores, pero no al revés.
- Si no se conoce los márgenes de trabajo de cada nivel, se recomienda utilizar el modo AUTO.

Parámetros de salida / acciones:

Configura la carga dinámica en modo de tensión constante.

9. Configurar parámetro “soft start”.

Función:

soft_start

Parámetros de entrada:

l_soft: Lista con la configuración “soft start”.

Ejemplo: [list channel {ch1 ch5} sst {5 ms}]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- sst: tiempo de alcance del régimen permanente {<value> mS | S... MIN | MAX}

Parámetros de salida / acciones:

Configura el parámetro “soft start” del isntrumento.

10. Configurar variación de corriente.

Función:

set_slew

Parámetros de entrada:

l_slew: Lista con la configuración de la variación de corriente por cada µs

Ejemplo: [list channel {ch1 ch5} slew 0.5 mode auto]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- slew: Amperios / µs { <value> }
- mode: módulo de corriente {AUTO | LOW | MED | HIGH}

Información adicional:

- El parámetro mode permite escoger el nivel de carga utilizado. Un módulo alto permite utilizar valores de módulos inferiores, pero no al revés.
- Si no se conoce los márgenes de trabajo de cada nivel, se recomienda utilizar el modo AUTO.

Parámetros de salida / acciones:

Configura la variación de corriente por cada µs.

11. Configurar un pulso | tren de pulsos de corriente.

Función:

set_pulse_train

Parámetros de entrada:

l_load_tran: Lista con la configuración del pulso / tren de pulsos.

Ejemplo (pulso):

[list channel {ch1 ch2} max {1 A} min {0.5 A} slew 0.01 mode {single}]

Ejemplo (tren de pulsos)

[list channel {ch1 ch2} max {1 A} min {0.5 A} slew 0.01 mode {train 5}]

Ejemplo (desactivar tren de pulsos)

[list channel {ch1 ch2} mode stop]

- channel: canales que serán comandados {CH1 CH2 CH3... ALL | NONE}
- max: valor máximo de corriente en cada pulso {<value > mA | A... MIN | MAX}
- min: valor mínimo de corriente en cada pulso {<value > mA | A... MIN | MAX}
- mode: selección del modo de pulso {{train freq_value } | single | stop}]
    - {freq_value}: frecuencia de pulso en Hz

Parámetros de salida / acciones:

Genera un pulso o un tren de pulsos de corriente.

12. Medir corriente de entrada | salida.

Función:

read_current

Parámetros de entrada:

channel: Canales cuya corriente se quiere medir. {CH1 CH2 CH3... ALL | NONE}

Ejemplo : {CH1 CH3}

Parámetros de salida / acciones:

Devuelve la medida de la corriente en Amperios

13. Medir tensión de entrada | salida.

Función:

read_voltage

Parámetros de entrada:

channel: Canales cuyo voltaje se quiere medir. {CH1 CH2 CH3... ALL | NONE}

Ejemplo : {CH1 CH3}

Parámetros de salida / acciones:

Devuelve la medida de la tensión en Voltios.

14. Medir potencia de entrada | salida.

Función:

read_power

Parámetros de entrada:

channel: Canales cuyo voltaje se quiere medir. {CH1 CH2 CH3... ALL | NONE}

Ejemplo : {CH1 CH3}

Parámetros de salida / acciones:

Devuelve la medida de la potencia en Watios.

15. Medir conductancia.

Función:

read_conductance

Parámetros de entrada:

channel: Canales cuyo voltaje se quiere medir. {CH1 CH2 CH3... ALL | NONE}

Ejemplo : {CH1 CH3}

Parámetros de salida / acciones:

Devuelve la medida de la conductancia en Siemens.

Capítulo 7

Osciloscopio: driver_rs232_tekttps2024

Librería destinada al comandado del osciloscopio Tektronix 2024.

Instanciación:

Parámetros del constructor:

serial_port: Puerto serie por el que está conectado el instrumento

Ejemplo: COM1

config_dir: Directorio donde se encuentran localizados los archivos de configuración del osciloscopio.

Ejemplo de instanciación:

```
set osci_config_dir "/oscilloscope_folder/"

set Osci [driver_rs232_tekttps2024 Osci "COM1" $osci_config_dir]
```

Funciones de la clase:

1. Guardar configuración actual.

Función:

getconfig

Parámetros de entrada:

file_name: Nombre que se le dará al fichero con la configuración del osciloscopio

Ejemplo: Config_1

Parámetros de salida / acciones:

Guarda la configuración actual del osciloscopio en un archivo.

2. Configurar osciloscopio con una configuración guardada.

Función:

setconfig

Parámetros de entrada:

file_name: Nombre que tendrá el fichero de configuración que será cargado.

Ejemplo: Config_1

Parámetros de salida / acciones:

Carga un archivo de configuración en el osciloscopio.

3. Guardar pantalla del osciloscopio.

Función:

hard_copy

Parámetros de entrada:

bmp_name: Nombre que se le dará a la imagen de la pantalla del osciloscopio.

Ejemplo: Imagen_1

Parámetros de salida / acciones:

Guarda la pantalla del osciloscopio en un archivo de imagen (.BMP)

4. Guardar forma de onda de un canal [Manual | Auto].

Función:

wave_curve

Parámetros de entrada:

wave_settings: Lista que contiene el modo de adquisición de la forma de onda.

Ejemplo: [list mode time] (Devuelve la escala de tiempos)

Ejemplo: [list mode auto channel 1] (Devuelve la forma de onda completa del canal 1)

Ejemplo: [list mode manual channel 1] (Devuelve la forma de onda que se encuentra entre los dos cursores del osciloscopio)

- mode: selecciona el modo de adquisición de la forma de onda o la escala de tiempos.
- channel: selecciona el canal del cual se extrae la forma de onda.

Parámetros de salida / acciones:

- Devuelve la forma de onda expresada en Voltios.
- Devuelve la escala de tiempos expresada en Segundos.

Capítulo 8

Funciones privadas presentes en instrumentos comandados por GPIB. (Programador)

Las siguientes funciones son comunes para todos los instrumentos que se comandan por GPIB. Cualquier función privada permanece oculta e inaccesible al usuario. Las funciones siguientes carecen de importancia a nivel de usuario, pero si se desea realizar cualquier cambio a bajo nivel es recomendable conocer su funcionamiento.

1. Instanciación del instrumento en el objeto gestor de errores (error_handle).

Función:

setup_error_handle

Parámetros de entrada:

Ninguno

Parámetros de salida / acciones:

Instancia, en el objeto de la clase "error_handle", el nombre y el puntero hacia el instrumento, permitiendo al objeto poder comandar el instrumento desde su misma clase.

2. Lectura de errores y máxima severidad asociada a los mismos.

Función:

error_report

Parámetros de entrada:

option: selecciona el parámetro que será devuelto por la función. { error | severity }

- error: devuelve una lista con todos los errores producidos.
- severity: de entre todos los niveles de severidad producidos, devuelve el valor más alto.

  El nivel de severidad asociado a un error se asignará en el diccionario de errores propio de cada instrumento.

Parámetros de salida / acciones:

Error: devuelve una lista con todos los errores producidos incluyendo el error nulo {+0, No Error}

Ejemplo: {-101, Invalid character} {-102, Syntax error } {+0, No Error}

Severity: devuelve un número entero correspondiente a la máxima severidad producida.

Ejemplo: 2

DOCUMENTO III

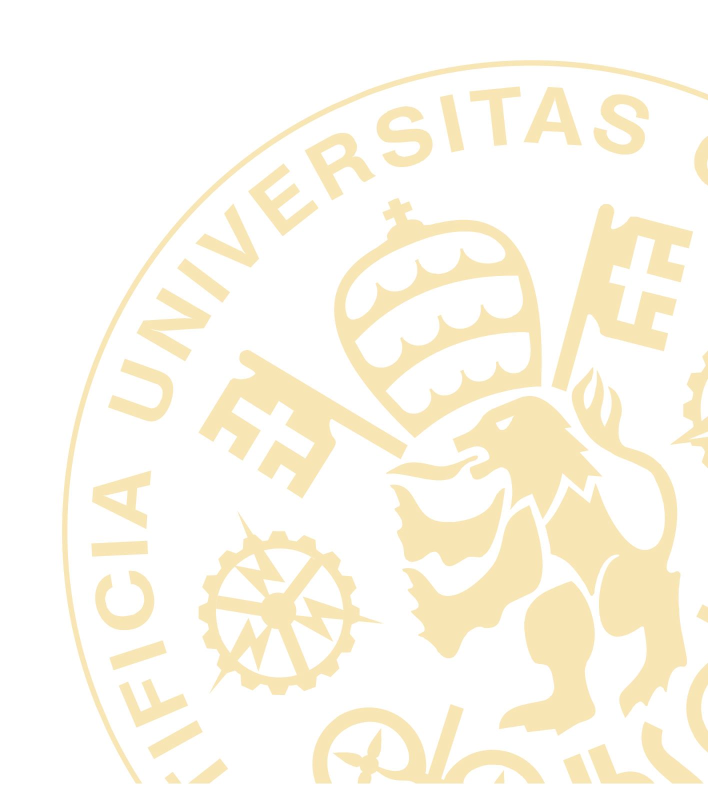
LIBRERIAS DEL SISTEMA



Índice general

PARTE I

DRIVER_GPIB



Librería de la clase

```
package provide driver_gpib 1.0

package require tclvisa

package require Itcl


namespace eval driver_gpib {

    ::itcl::class driver_gpib {

    ## \private
    private variable aux
        ## \private
    private variable num_dev
        ## \private
    private variable _id
        ## \private
    private variable num
        ## \private
    private variable id
        ## \private
    private variable y
        ## \private
    private variable end
        ## \private
    private variable flag_nc
        ## \private
    private variable status
        ## \private
    private variable fp
        ## \private
    private variable list_devices


    ## Initialize the parameters of the class
    #
    # @param d_log_name (list): List with the path and the name of the file that will be used to register all commands sent
    # - \b Fist element: File path
    # - \b Second element: Name of file
    # @param  list_devices (list): List with the instruments that are going to be used in the test secquence.
    # - \b First element of one list: Name
    # - \b Second element of the list: Bus connection
    # - \b Thrird element of the list: Bus address
        constructor { d_log_name _list_devices} {
            #Set Address and name in different variables
            set file_addrs [lindex $d_log_name 0]
            set file_name [lindex $d_log_name 1]
            #Open file in which the log is going to be written down
            set log_add [file join [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) $file_addrs $file_name\.log]]
            set fp [open $log_add a+]

            #Set Instrument list path
            set list_devices $_list_devices
        }

        destructor  {
            #Close file when test's sequence ends
            close $fp
        }


    ## Method to read all devices connected to the computer
    #
    # @param verbose (boolean): If true, the elements will be shown individually. If false, the elements will be returned as a list of
    values with the different measures.
    # @return (string): Just in no verbose mode,
        public method get_devices_connected {{verbose true}} {
            set num_dev 100
            set list_adr [list]
            set rm [visa::open-default-rm]
            for { set x 0 } { $x < $num_dev } { incr x } {
                set aux "GPIB0::$x\::INSTR"

                if { [catch { set vi [visa::open $rm $aux] } rc] } {

                } else {
                #Read device ID
                puts $vi "*IDN?"

                #Remove useless part of ID
                set _id [gets $vi]
                #Characters until second ","
                set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
                #Redefine device ID
                set id [string range $_id 0 $num]

                lappend list_adr "$aux gpib $id"
                }
            }

            if {[llength $list_adr]==0}  {
                puts "No equipment connected"
            }

            if { $verbose == "true" } {
                foreach gpib_device $list_adr {
                    puts $gpib_device
                }

                return ""
            }

            return $list_adr

        }




      ## Method to check if all devices declared in the list are currecntly corrected.
      #
      # @return (boolean): If 0: Some devices are not connected. If 1: All devices are connected
        public method check_devices_connected {} {
                #Path / Source Device File

                #Number of devices that will be checked
                set y [llength $list_devices]
                #Bucle for writing
                for { set x 0 } { $x < $y} { incr x } {
            #Initialize flag not connected
                set flag_nc 0
```

```
                #Check if instrument is connected
                array set instrument [lindex $list_devices  $x]
                if { [catch { set rm [visa::open-default-rm] } rc] } {
                        set flag_nc 1
                } else {
                        if { [catch { set vi [visa::open $rm $instrument(visa_addrs)] } rc ] } {
                        set flag_nc 1
                    } else {
                        set aux "$instrument(visa_addrs)"
                        if { [catch { puts $vi "*IDN?" } rc] } {
                                set flag_nc 1
                        } else {
                        #Visa Address Right
                        #Check if name corresponds with name in list
                                set _id [gets $vi]
                                set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
                                set id_name [string range $_id 0 $num]

                                if { $instrument(name) == $id_name } {
                                   set flag_nc 0
                                } else {
                                   set flag_nc 1
                                }
                        }
                        }
                }

                if { $x == [expr ($y -1)]} {
                     if { $flag_nc == 1} {
                        return 0
                     } else {
                        return 1
                     }
                }
               }
        }



    ## Method to write in a file:  time, device name, device addres, and command sent.
        # @param dev_name (string): Contains the name of the device connected
        # @param dev_addrs (string): Contains the gpib address of the device
        # @param _command (string): command sent to the device
        public method command_sent_wfile { dev_name dev_addrs _command} {
            set command $_command

            #DATE:
            set date [clock format [clock second] -format %D]
            set time [clock format [clock second] -format %T]

            #COMMAND LOG: Write in the file the "date" "time" "Device address" "Command sent"
            puts $fp "$date $time $dev_addrs \"$command\""

        }

  }
  namespace export driver_gpib
}
```

PARTE II

ERROR_HANDLE



Librería de la clase

```
package provide error_handle 1.0

namespace eval error_handle {
   namespace import ::itcl::*

   ::itcl::class error_handle {

        #Variables
        private variable instrument_list
        private variable sequence_controlled_exit
        private variable crlogger

        #Constructor
        constructor { _crlogger} {
            set instrument_list [list]
            set crlogger $_crlogger
        }


        public method error_report_screen { dev_addrs error_data} {
            #ERROR LOG
            #ERROR STRING LENGTH
            set z [llength $error_data]

            for { set w 0 } { $w < [expr ($z-1)]} { incr w } {
                set val [expr [string first "," [lindex $error_data $w] 0]]
                lappend error_val [string range [lindex $error_data $w] 1 [expr ($val-1)]]

                if {[expr [lindex $error_val $w]] != 0} {
                    # puts stdout "$dev_addrs [lindex $error_data $w]"; #Error message through screen
                    $crlogger rep_error "$dev_addrs [lindex $error_data $w]"; #Error message through test logger
                }
            }
        }


        public method execute_controlled_exit { severity } {
            # array set in $instrument_list

            foreach {id ref} $instrument_list {
                eval "set $id $ref"
            }

            if {$severity < 10 && $severity > 0 } {
                puts "error leve producido"
            }

            if { $severity > 10 } {
                puts "error grave producido"

                foreach step $sequence_controlled_exit {
                    eval $step
                }

                error $severity
            }

        }


        public method load_controlled_exit { controlled_exit } {
            set sequence_controlled_exit $controlled_exit
        }

        public method add_instrument { instrument_id instr_ref } {
            lappend instrument_list $instrument_id  $instr_ref
            puts $instrument_list
        }



   }
    namespace export error_handle
}
```

PARTE III

DRIVER_GPIB_AGIL34970A



Librería de la clase

```
package provide driver_gpib_agil34970a 1.0
package require class_scanner
package require Itcl


namespace eval driver_gpib_agil34970a {

    ::itcl::class driver_gpib_agil34970a {
        inherit ::class_scanner::scanner

        ## \private
        private variable visaAddr
        ## \private
        private variable vi
        ## \private
        private variable rc
        ## \private
        private variable rm
        ## \private
        private variable device_name
        ## \private
        private variable err
        ## \private
        private variable gpib
        ## \private
        private variable dict_err
        ## \private
        private variable err_dict_output
        ## \private
        private variable c_err_msg
        ## \private
        private variable connection_error_value
        ## \private
        private variable con_def_msg
        ## \private
        private variable con_def_msg_er
        ## \private
        private variable severity_nu
        ## \private
        private variable max
        ## \private
        private variable logger


        ## Initializate the parameters of the class
        #
        # - \b Constructor function:
        #    + \b Stablish a GPIB connection through the GPIB port indicated
        #    + \b Sets the error dictionary
        #    + \b Sets the severity connection value
        #    + \b Adds the instrument reference in the error_handle object
        #
        # @param _visa_addrs (string): instrument GPIB port
        # @param _gpib (reference): Class reference to driver_gpib object
        # @param _logger (reference): Class reference to CRLogger object
        constructor { _visa_addrs _gpib  {_logger ::logger::CRLogger}  } {
            set logger $_logger
            #Pointer to driver_gpib
            set gpib $_gpib

            # open device
            set visaAddr $_visa_addrs
            # get handle to default resource manager
            if { [catch { set rm [visa::open-default-rm] } rc] } {
                $logger rep_error "Error opening default resource manager\n$rc"
            } else {
                set rm [visa::open-default-rm]
            }

            # check if devide opened
            if { [catch { set vi [visa::open $rm $visaAddr] } rc] } {
                puts "Error opening instrument `$visaAddr`\n$rc"
            } else {
                set vi [visa::open $rm "$visaAddr"]
                # Set proper timeout
                fconfigure $vi -timeout 500
            }

            # Get ID from instrument
            puts $vi "*IDN?"
            #Remove useless part of ID
            set _id [gets $vi]
            #Characters until second ","
            set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
            #Redefine device ID
            set id [string range $_id 0 $num]
            #Device name
            set device_name $id

            #Setup error dictionary
            set fp [open [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) setup sw source tcl
            driver_gpib_agil34970a driver_gpib_agil34970a_dict_error.json] r]
            set dict_err [read $fp]
            set dict_err [json::json2dict $dict_err]

            # SEVERITY CONNECTION ERROR VALUE
            set severity_nu 2
            set connection_error_value -1
            set c_err_msg [list "$connection_error_value, \"Connection error\" $severity_nu" "+0, \"No Error\" 0"]
            set con_def_msg "error writing \"$vi\": Unknown error"

            #Specific error report when disconection
            set c_err_msg_er "$connection_error_value, \"Connection error\" $severity_nu"

            #Add instrument in error_handle
            $this setup_error_handle

            #Initialize maximum severity
            set max 0
        }

        ## Method to change the display state:
        #  @param on_off (string) : "on" (Turns on the display)  "off" (Turns off the display)
        public method display { on_off } {
            #-----------------------------------------------------------------------------------------------------
            if { [catch {
            #-----------------------------------------------------------------------------------------------------
                    puts $vi "DISPlay $on_off"
                    $gpib command_sent_wfile $device_name $visaAddr "DISPlay $on_off"
                    $error_handle error_report_screen $visaAddr [$this error_report error_message]
            #-----------------------------------------------------------------------------------------------------
```

```
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]
}

## Method to read DC voltage
# @param v_conf_dc (parameter_list) list with the following format:
# - \b Example: [list channel {105:110,215} range {5 mV}  resolution {1 mV}]
#    + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
    #    + \b range  (string): Range of the measure  {<value> V|mV ...|MIN|MAX|DEF}
#    + \b resolution (string): Resolution of the measure   {<value> V|mV ...|MIN|MAX|DEF}
# @return (integer): Voltage measured (V)
public method read_voltage_dc { v_conf_dc} {
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    array set v_param $v_conf_dc
    puts $vi "MEAS:VOLT:DC? $v_param(range),$v_param(resolution),(@$v_param(channel))"
    set id [gets $vi]
    $gpib command_sent_wfile $device_name $visaAddr "MEAS:VOLT:DC?  $v_param(range),$v_param(resolution),(@$v_param(channel))"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    #------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------------------
    return $id
    #------------------------------------------------------------------------------------------------------------
    }
}


## Method to read AC voltage
# @param v_conf_ac (parameter_list) list with the following format:
# - \b Example: [list channel {105:110,215} range {5 mV}  resolution {1 mV}]
#    + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
    #    + \b range  (string): Range of the measure  {<value> V|mV ...|MIN|MAX|DEF}
#    + \b resolution (string): Resolution of the measure   {<value> V|mV ...|MIN|MAX|DEF}
# @return (integer): Voltage measured (V)
public method read_voltage_ac  { v_conf_ac} {
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    array set v_param $v_conf_ad
    puts $vi "MEAS:VOLT:AC?  $v_param(range),$v_param(resolution),(@$v_param(channel))"
    set id [gets $vi]
    $gpib command_sent_wfile $device_name $visaAddr "MEAS:VOLT:AC?  $v_param(range),$v_param(resolution),(@$v_param(channel))"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    #------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------------------
    return $id
    #------------------------------------------------------------------------------------------------------------
    }
}

## Method to read DC current
# @param c_conf_dc (parameter_list) list with the following format:
# - \b Example: [list channel {105:110,215} range {200 mA} resolution {1 mA}]
#    + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
    #    + \b range  (string): Range of the measure  {<value> A|mA ...|MIN|MAX|DEF}
#    + \b resolution (string): Resolution of the measure   {<value> A|mA ...|MIN|MAX|DEF}
# @return (Integer): Current measured (A)
public method read_current_dc { c_conf_dc } {
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    array set c_param $c_conf_dc
    puts $vi "MEAS:CURR:DC? $c_param(range),$c_param(resolution),(@$c_param(channel))"
    $gpib command_sent_wfile $device_name $visaAddr "MEAS:CURR:DC? $c_param(range),$c_param(resolution),(@$c_param(channel))"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    set id [gets $vi]
    #------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------------------
    return $id
    #------------------------------------------------------------------------------------------------------------
    }

}

## Method to read AC current
# @param c_conf_ac (parameter_list) list with the following format:
# - \b Example: [list channel {105:110,215} range {200 mA} resolution {1 mA}]
#    + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
    #    + \b range  (string): Range of the measure  {<value> A|mA ...|MIN|MAX|DEF}
#    + \b resolution (string): Resolution of the measure   {<value> A|mA ...|MIN|MAX|DEF}
# @return (Integer): Current measured (A)
public method read_current_ac { c_conf_ac } {
```

```
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    array set c_param $c_conf_dc
    puts $vi "MEAS:CURR:AC $c_param(range), $c_param(resolution),(@$c_param(channel))"
    set id [gets $vi]
    $gpib command_sent_wfile $device_name $visaAddr "MEAS:CURR:AC $c_param(range), $c_param(resolution),(@$c_param(channel))"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    #------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]
    return $id
}

## Method to read 2 wires resistances
# @param res_conf (parameter_list) list with the following format:
# - \b Example: [list channel {105:110,215} range {2 ohm} resolution {1 mohm}]
#   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
    #   + \b range  (string): Range of the measure  {<value> ohm|kohm ...|MIN|MAX|DEF}
#   + \b resolution (string): Resolution of the measure   {<value> ohm|kohm ...|MIN|MAX|DEF}
# @return (Integer): Resistance measured (Ohm)
public method read_resistancex2 {res_conf} {
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    array set res_param $res_conf
    puts $vi "MEAS:RES? $res_param(range),$res_param(resolution),(@$res_param(channel))"
    set id [gets $vi]
    $gpib command_sent_wfile $device_name $visaAddr "MEAS:RES? $res_param(range),$res_param(resolution),(@$res_param(channel))"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    #------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------------------
    return $id
    #------------------------------------------------------------------------------------------------------------
    }
}

## Method to read 4 wires resistances
# @param res_conf (parameter_list) list with the following format:
# - \b Example: [list channel {105:110,215} range {2 ohm} resolution {1 mohm}]
#   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
    #   + \b range  (string): Range of the measure  {<value> ohm|kohm ...|MIN|MAX|DEF}
#   + \b resolution (string): Resolution of the measure   {<value> ohm|kohm ...|MIN|MAX|DEF}
# @return (Integer): Resistance measured (Ohm)
public method read_resistancex4 {res_conf} {
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    array set res_param $res_conf
    puts $vi "MEAS:FRES? $res_param(range),$res_param(resolution),(@$res_param(channel))"
    set id [gets $vi]
    $gpib command_sent_wfile $device_name $visaAddr "MEAS:FRES? $res_param(range),$res_param(resolution),(@$res_param(channel))"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------------------
    return $id
    #------------------------------------------------------------------------------------------------------------
    }
}

## \private
# Method that setup the instrument in an error_handle object previusly declared
# No parameters needed
private method setup_error_handle {} {
    set possition [string last "::" $this]
    set dev_name [string range $this  [expr $possition+2] end]

    $error_handle add_instrument $dev_name $this
}

## \private
## Method that returns the errors message or severty produced during the sequence
# @param option string with the output option (error | severity)
# @return (parameter_list) list with all the errors or max severity produced
# - \b Example (error): {-101  "Sintax error"} {+0 "No error"}
#   + \b First value: Error Code
#   + \b Second value: Error message
# - \b Example (severity): +5
private method error_report { option } {
    #------------------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------------------
    set err_dict_output [list]
    set err_sev_list [list]


    puts $vi "SYST:ERR?"
    set err [gets $vi]
    set val [expr [string first ","  $err 0]]
    set error_val [string range $err 0 [expr ($val-1)]]


    if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
```

```
                    lappend err_dict_output "$err"
                    lappend err_sev_list 1
            } else {
                set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                lappend err_dict_output "$error_val,  \"$definition\""
                lappend err_sev_list $severity_val
            }

            for {set x 0} { $err != "+0,\"No error\"" } {incr x} {
                puts $vi "SYST:ERR?"
                set err [gets $vi]

                set val [expr [string first ","  $err 0]]
                set error_val [string range $err 0 [expr ($val-1)]]



                if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                    lappend err_dict_output "$err"
                    lappend err_sev_list 1
                } else {
                    set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                    lappend err_dict_output "$error_val,  \"$definition\""
                    lappend err_sev_list $severity_val

                }
            }



            foreach n $err_sev_list {
                if {$n > $max} {
                    set max $n
                }
            }

            set error_message $err_dict_output
            set severity $max

            #--------------------------------------------------------------------------------------------------------------
            } err_code]} {
                if {$err_code == $con_def_msg} {
                    $error_handle  execute_controlled_exit $severity_nu
                }
            } else {
            #--------------------------------------------------------------------------------------------------------------
            if { $severity == 0 && $option == "severity"} {
            } else {
                if {$severity != 0 && $option == "severity"} {
                    set max 0
                }
                    eval return $$option
                }
            }
        }
    }

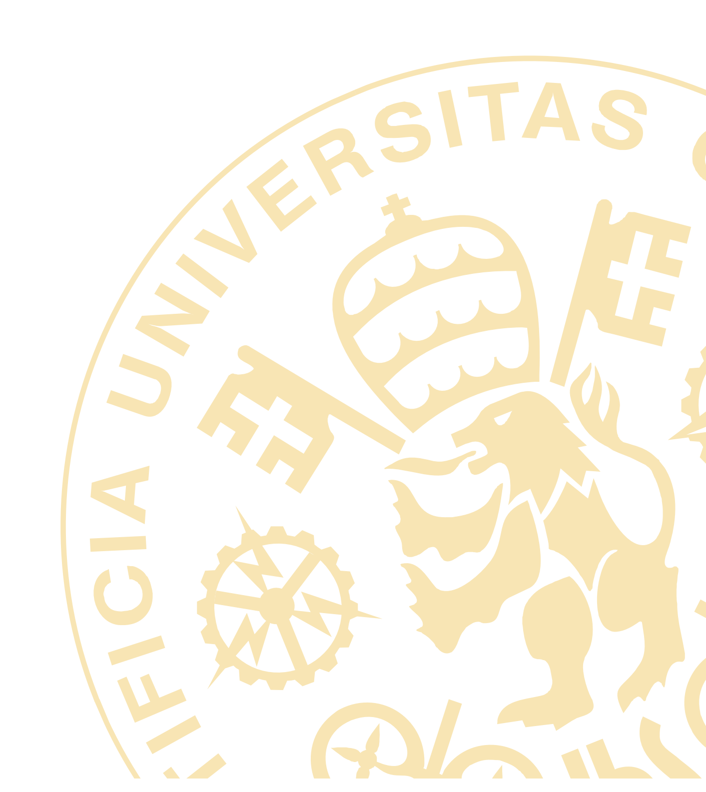
     namespace export driver_gpib_agil34970a
}
```

Dicionario de errores

```
{
"+0":   {"No Error"                                                                           :"0"},
"-100": {"Command error (generic command error)"                                              :"0"},
"-101": {"Invalid character"                                                                  :"0"},
"-102": {"Syntax error (unrecognized command or data type)"                                   :"0"},
"-103": {"Invalid separator (illegal character encountered in place of separator)"            :"0"},
"-104": {"Data type error (e.g., “numeric or string expected, got block date”)"               :"0"},
"-105": {"GET not allowed ( <GET> inside a program message)"                                  :"0"},
"-108": {"Parameter not allowed (too many parameters)"                                        :"0"},
"-109": {"Missing parameter (too few parameters)"                                             :"0"},
"-112": {"Program mnemonic too long (maximum 12 characters)"                                  :"0"},
"-113": {"Undefined header (syntactical correct but not defined for this device)"             :"0"},
"-114": {"Header suffix out of range"                                                         :"0"},
"-121": {"Invalid character in number (e.g. alpha in decimal data, etc.)"                     :"0"},
"-123": {"Exponent too large ( numeric overflow; exponent magnitude >32000)"                  :"0"},
"-124": {"Too many digits (number too long; more than 255 digits received)"                   :"0"},
"-128": {"Numeric data not allowed (numeric data not accepted where positioned)"              :"0"},
"-131": {"Invalid suffix (unrecognized suffix, or suffix not appropriate)"                    :"24"},
"-134": {"Suffix too long"                                                                    :"0"},
"-138": {"Suffix not allowed (numeric element does not allow suffixes)"                       :"0"},
"-141": {"Invalid character data (bad character, or unrecognized)"                            :"0"},
"-144": {"Character data too long (maximum length is 12 characters)"                          :"0"},
"-148": {"Character data not allowed (character data not accepted where positioned)"          :"0"},
"-150": {"String data error (generic string error)"                                           :"0"},
"-151": {"Invalid string data (e.g., END received before close quote)"                        :"0"},
"-158": {"String data not allowed (string data not accepted where positioned)"                :"0"},
"-160": {"Block data error (generic data block error)"                                        :"0"},
"-161": {"Invalid block data (e.g., END received before length satisfied)"                    :"0"},
"-168": {"Block data not allowed (block data not accepted where positioned)"                  :"0"},
"-178": {"Expression data not allowed"                                                        :"0"},
"-211": {"Trigger ignored"                                                                    :"0"},
"-213": {"INIT ignored"                                                                       :"0"},
"-214": {"Trigger deadlock"                                                                   :"0"},
"-220": {"Parameter error"                                                                    :"0"},
"-221": {"Settings conflict (uncoupled parameters)"                                           :"0"},
"-222": {"Data out of range (e.g., outside the range of this device)"                         :"1"},
"-223": {"Too much data (out of memory; block, string, or expression too long)"               :"0"},
"-224": {"Illegal parameter value"                                                            :"0"},
"-230": {"Data stale"                                                                         :"0"},
"-240": {"Hardware error (device-dependent)"                                                  :"0"},
"-241": {"Hardware missing (device-dependent)"                                                :"0"},
"-310": {"System error (device-dependent)"                                                    :"0"},
"-313": {"Calibration memory lost (out of calibration due to memory failure)"                 :"0"},
"-330": {"Self-test failed (more specific data after “;”)"                                    :"0"},
"-350": {"Queue overflow (errors lost due to too many errors in queue)"                       :"0"},
"-400": {"Query error (generic query error)"                                                  :"0"},
"-410": {"Query INTERRUPTED (query followed by DAB or GET before response complete)"          :"0"},
"-420": {"Query UNTERMINATED (addressed to talk, incomplete programming message received)"    :"0"},
"-430": {"Query DEADLOCKED (too many queries in command string)"                              :"0"},
"-440": {"Query UNTERMINATED (query received after query for indefinite response)"            :"0"},
"111": {"Channel list: slot number out of range"                                              :"0"},
"112": {"Channel list: channel number out of range"                                           :"0"},
"113": {"Channel list: empty scan list"                                                       :"0"},
"201": {"Memory lost: stored state"                                                           :"0"},
"202": {"Memory lost: power-on state"                                                         :"0"},
"203": {"Memory lost: stored readings"                                                        :"0"},
"204": {"Memory lost: time and date"                                                          :"0"},
"221": {"Settings conflict: calculate limit state forced off"                                 :"0"},
"222": {"Settings conflict: module type does not match stored state"                          :"0"},
"223": {"Settings conflict: trig source changed to IMM"                                       :"0"},
"224": {"Settings conflict: chan adv source changed to IMM"                                   :"0"},
"225": {"Settings conflict: DMM disabled or missing"                                          :"0"},
"226": {"Settings conflict: DMM enabled"                                                      :"0"},
"251": {"Unsupported temperature transducer type"                                             :"0"},
"261": {"Not able to execute while scan initiated"                                            :"0"},
"271": {"Not able to accept unit names longer than 3 characters"                              :"0"},
"272": {"Not able to accept character in unit name"                                           :"0"},
"281": {"Not able to perform on more than one channel"                                        :"0"},
"291": {"Not able to recall state: it is empty"                                               :"0"},
"292": {"Not able to recall state: DMM enable changed"                                        :"0"},
"301": {"Module currently committed to scan"                                                  :"0"},
"303": {"Module not able to perform requested operation"                                      :"0"},
"305": {"Not able to perform requested operation"                                             :"0"},
"306": {"Part of a 4-wire pair"                                                               :"0"},
"307": {"Incorrectly configured ref channel"                                                  :"0"},
"501": {"I/O processor: isolator framing error"                                               :"0"},
"502": {"I/O processor: isolator overrun error"                                               :"0"},
"511": {"Communications: RS-232 framing error"                                                :"0"},
"512": {"Communications: RS-232 overrun error"                                                :"0"},
"513": {"Communications: RS-232 parity error"                                                 :"0"},
"514": {"RS-232 only: unable to execute using HP-IB"                                          :"0"},
"521": {"Communications: input buffer overflow"                                               :"0"},
"522": {"Communications: output buffer overflow"                                              :"0"},
"532": {"Not able to achieve requested resolution"                                            :"0"},
"540": {"Not able to null channel in overload"                                                :"0"},
"550": {"Not able to execute command in local mode"                                           :"0"},
"601": {"Self-test: front panel not responding"                                               :"0"},
"602": {"Self-test: RAM read/write"                                                           :"0"},
"603": {"Self-test: A/D sync stuck"                                                           :"0"},
"604": {"Self-test: A/D slope convergence"                                                    :"0"},
"605": {"Self-test/Cal: not able to calibrate rundown gain"                                   :"0"},
"606": {"Self-test/Cal: rundown gain out of range"                                            :"0"},
"607": {"Self-test: rundown too noisy"                                                        :"0"},
"608": {"Self-test: serial configuration readback"                                            :"0"},
"609": {"Self-test: DC gain x1"                                                               :"0"},
"610": {"Self-test: DC gain x10"                                                              :"0"},
"611": {"Self-test: DC gain x100"                                                             :"0"},
"612": {"Self-test: Ohms 500 nA source"                                                       :"0"},
"613": {"Self-test: Ohms 5 uA source"                                                         :"0"},
"614": {"Self-test: DC 300V zero"                                                             :"0"},
"615": {"Self-test: Ohms 10 uA source"                                                        :"0"},
"616": {"Self-test: DC current sense"                                                         :"0"},
"617": {"Self-test: Ohms 100 uA source"                                                       :"0"},
"618": {"Self-test: DC high voltage attenuator"                                               :"0"},
"619": {"Self-test: Ohms 1 mA source"                                                         :"0"},
"620": {"Self-test: AC rms zero"                                                              :"0"},
"621": {"Self-test: AC rms full scale"                                                        :"0"},
"622": {"Self-test: frequency counter"                                                        :"0"},
"623": {"Self-test: not able to calibrate precharge"                                          :"0"},
"624": {"Self-test: not able to sense line frequency"                                         :"0"},
"625": {"Self-test: I/O processor not responding"                                             :"0"},
"626": {"Self-test: I/O processor self-test"                                                  :"0"},
"701": {"Cal: security disabled by jumper"                                                    :"0"},
"702": {"Cal: secured"                                                                        :"0"},
"703": {"Cal: invalid secure code"                                                            :"0"},
"704": {"Cal: secure code too long"                                                           :"0"},
"705": {"Cal: aborted"                                                                        :"0"},
"706": {"Cal: value out of range"                                                             :"0"},
"707": {"Cal: signal measurement out of range"                                                :"0"},
"708": {"Cal: signal frequency out of range"                                                  :"0"},
"709": {"Cal: no cal for this function or range"                                              :"0"},
"710": {"Cal: full scale correction out of range"                                             :"0"},
"720": {"Cal: DCV offset out of range"                                                        :"0"},
"721": {"Cal: DCI offset out of range"                                                        :"0"},
"722": {"Cal: RES offset out of range"                                                        :"0"},
"723": {"Cal: FRES offset out of range"                                                       :"0"},
"724": {"Cal: extended resistance self cal failed"                                            :"0"},
"725": {"Cal: 300V DC correction out of range"                                                :"0"},
"730": {"Cal: precharge DAC convergence failed"                                               :"0"},
"731": {"Cal: A/D turnover correction out of range"                                           :"0"},
"732": {"Cal: AC flatness DAC convergence failed"                                             :"0"},
"733": {"Cal: AC low frequency convergence failed"                                            :"0"},
"734": {"Cal: AC low frequency correction out of range"                                       :"0"},
"735": {"Cal: AC rms converter noise correction out of range"                                 :"0"},
"736": {"Cal: AC rms 100th scale correction out of range"                                     :"0"},
"740": {"Cal data lost: secure state"                                                         :"0"},
"741": {"Cal data lost: string data"                                                          :"0"},
"742": {"Cal data lost: DCV corrections"                                                      :"0"},
"743": {"Cal data lost: DCI corrections"                                                      :"0"},
"744": {"Cal data lost: RES corrections"                                                      :"0"},
"745": {"Cal data lost: FRES corrections"                                                     :"0"},
"746": {"Cal data lost: AC corrections"                                                       :"0"},
"747": {"Config data lost: HP-IB address"                                                     :"0"},
"748": {"Config data lost: RS-232"                                                            :"0"},
"749": {"DMM relay count data lost"                                                           :"0"},
"901": {"Module hardware: unexpected data received"                                           :"0"},
"902": {"Module hardware: missing stop bit"                                                   :"0"},
"903": {"Module hardware: data overrun"                                                       :"0"},
"904": {"Module hardware: protocol violation"                                                 :"0"},
"905": {"Module hardware: early end of data"                                                  :"0"},
"906": {"Module hardware: missing end of data"                                                :"0"},
"907": {"Module hardware: module srq signal stuck low"                                        :"0"},
"908": {"Module hardware: not responding"                                                     :"0"},
"910": {"Module reported an unknown module type"                                              :"0"},
"911": {"Module reported command buffer overflow"                                             :"0"},
"912": {"Module reported command syntax error"                                                :"0"},
"913": {"Module reported nonvolatile memory fault"                                            :"0"},
"914": {"Module reported temperature sensor fault"                                            :"0"},
"915": {"Module reported firmware defect"                                                     :"0"},
"916": {"Module reported incorrect firmware installed"                                        :"0"}
}
```

PARTE IV

DRIVER_GPIB_HP6653A



Librería de la clase

```
package provide driver_gpib_hp6653a 1.0
package require class_powersupply
package require Itcl


namespace eval driver_gpib_hp6653a {

     ::itcl::class driver_gpib_hp6653a {
        inherit ::class_powersupply::powersupply

        ## \private
        private variable visaAddr
        ## \private
        private variable on_off
        ## \private
        private variable vi
        ## \private
        private variable rc
        ## \private
        private variable rm
        ## \private
        private variable device_name
        ## \private
        private variable err
        ## \private
        private variable gpib
        ## \private
        private variable dict_err
        ## \private
        private variable err_dict_output
        ## \private
        private variable c_err_msg
        ## \private
        private variable connection_error_value
        ## \private
        private variable con_def_msg
        ## \private
        private variable con_def_msg_er
        ## \private
        private variable severity_nu
        ## \private
        private variable err_sev_list
        ## \private
        private variable max
        ## \private
        private variable error_handle
        ## \private


        ## Initializate the parameters of the class
        #
        # - \b Constructor function:
        #   + \b Stablish a GPIB connection through the GPIB port indicated
        #   + \b Sets the error dictionary
        #   + \b Sets the severity connection value
        #   + \b Adds the instrument reference in the error_handle object
        #
        # @param _visa_addrs (string): instrument GPIB port
        # @param _gpib (reference): Class reference to driver_gpib object
        # @param _logger (reference): Class reference to CRLogger object
            constructor { _visa_addrs _gpib d_error_handle} {
            #Pointer to $$error_handle class
            set error_handle $d_error_handle

            #Pointer to driver_gpib
            set gpib $_gpib

            # open device
            set visaAddr $_visa_addrs
            # get handle to default resource manager
            if { [catch { set rm [visa::open-default-rm] } rc] } {
                puts stderr "Error opening default resource manager\n$rc"
            } else {
                set rm [visa::open-default-rm]
            }

            # check if devide opened
            if { [catch { set vi [visa::open $rm $visaAddr] } rc] } {
                puts "Error opening instrument `$visaAddr`\n$rc"
            } else {
                set vi [visa::open $rm "$visaAddr"]
                # Set proper timeout
                fconfigure $vi -timeout 500
            }

            # Get ID from instrument
            puts $vi "*IDN?"
            #Remove useless part of ID
            set _id [gets $vi]
            #Characters until second ","
            set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
            #Redefine device ID
            set id [string range $_id 0 $num]
            #Device name
            set device_name $id

            #Setup error dictionary
            set fp [open [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) setup sw source tcl driver_gpib_hp6653a
             driver_gpib_hp6653a_dict_error.json] r]
            set dict_err [read $fp]
            set dict_err [json::json2dict $dict_err]

            # SEVERITY CONNECTION ERROR VALUE
            set severity_nu 2
            set connection_error_value -1
            set c_err_msg [list "$connection_error_value, \"Connection error\" $severity_nu" "+0, \"No Error\" 0"]
            set con_def_msg "error writing \"$vi\": Unknown error"

            #Specific error report when disconection
            set c_err_msg_er "$connection_error_value, \"Connection error\" $severity_nu"

            #Add instrument in error_handle
            $this setup_error_handle

            #Initialize maximum severity
            set max 0
            }

         ## Method to change the display state:
         #  @param on_off (string) : "on" (Turns on the display)  "off" (Turns off the display)
         public method display { on_off } {
                #-----------------------------------------------------------------------------------------------------------
```

```
        if { [catch {
        #---------------------------------------------------------------------------------------------------------
                puts $vi "DISPlay $on_off"
                $gpib command_sent_wfile $device_name $visaAddr "DISPlay $on_off"
                $error_handle error_report_screen $visaAddr [$this error_report error_message]
        #---------------------------------------------------------------------------------------------------------
        } err_code]} {
            if {$err_code == $con_def_msg} {
                $error_handle error_report_screen $visaAddr $c_err_msg
            } else {
                $error_handle error_report_screen $visaAddr "Unknown error"
            }
        }
        #---------------------------------------------------------------------------------------------------------
        $error_handle execute_controlled_exit [$this error_report severity]
 }


## Method to enable / disable power supply output
# @param output_conf (boolean): If "on" the power supply will switch on. If "off" the power supply will switch off
public method output { output_conf } {
        #---------------------------------------------------------------------------------------------------------
        if { [catch {
        #---------------------------------------------------------------------------------------------------------

        set on_off $output_conf

        puts $vi "OUTP $output_conf"
        $gpib command_sent_wfile $device_name $visaAddr "OUTP $output_conf"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        #---------------------------------------------------------------------------------------------------------
        } err_code]} {
            if {$err_code == $con_def_msg} {
                $error_handle error_report_screen $visaAddr $c_err_msg
            } else {
                $error_handle error_report_screen $visaAddr "Unknown error"
            }
        }
        #---------------------------------------------------------------------------------------------------------
        $error_handle execute_controlled_exit [$this error_report severity]

 }


 ## Method to configure voltage and current level without changing the state of the output.
 #  @param output_vc (parameter_list) list with the following format:
 # - \b Example: [list voltage {50 mV} current {3 mA}]
 #   + \b Voltage (string): Indicates voltage level {<value> V|mV...MIN|MAX}
 #   + \b Current (string): Indicates current level {<value> A|mA...MIN|MAX}]
 public method set_ps_vc { output_vc } {
        $this set_psv $output_vc(voltage)
        $this set_psc $output_vc(current)
 }


## Configure voltage (Without changing output state)
#  @param output_v (parameter_list) list with the following format:
# - \b Example: [list voltage {50 mV}
#   + \b Voltage (string): Indicates voltage level {<value> V|mV...MIN|MAX}
public method set_psv { output_v } {
    #---------------------------------------------------------------------------------------------------------
    if { [catch {
    #---------------------------------------------------------------------------------------------------------
    array set o_v  $output_v
    puts $vi "VOLT $o_v(voltage)"
    $gpib command_sent_wfile $device_name $visaAddr "VOLT $o_v(voltage)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    #---------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #---------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
 }


## Configure maximun level of current (Without changing output state)
#  @param output_c (parameter_list) list with the following format:
# - \b Example: [list current {100 mV}
#   + \b current (string): Indicates current level {<value> A|mA...MIN|MAX}
public method set_psc { output_c } {
        #---------------------------------------------------------------------------------------------------------
        if { [catch {
        #---------------------------------------------------------------------------------------------------------
        array set o_c  $output_c
        puts $vi "CURR $o_c(current)"
        $gpib command_sent_wfile $device_name $visaAddr "CURR $o_c(current)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        #---------------------------------------------------------------------------------------------------------
        } err_code]} {
            if {$err_code == $con_def_msg} {
                $error_handle error_report_screen $visaAddr $c_err_msg
            } else {
                $error_handle error_report_screen $visaAddr "Unknown error"
            }
        }
        #---------------------------------------------------------------------------------------------------------
        $error_handle execute_controlled_exit [$this error_report severity]
 }

 ## Method to configure voltage and current protection (Voltage is set at v_level) (Without changing output state)
 # @param protect_vc (parameter_list) list with the following format:
 # - \b Example: [list v_level {70 mV}   overv_level {1 V}   c_level {60 mA}]
 #   + \b v_level (string): Output voltage value  {<value> V|mV...MIN|MAX}
 #   + \b overv_level (string): Protection voltage level {<value> V|mV...MIN|MAX}
 #   + \b c_level (string): Protection current level {<value> A|mA...MIN|MAX}
 public method set_prot_psvc { protect_vc } {
        #---------------------------------------------------------------------------------------------------------
        if { [catch {
        #---------------------------------------------------------------------------------------------------------
        array set p_vc $protect_vc
        puts $vi "VOLT:PROT $p_vc(v_level)"
        $gpib command_sent_wfile $device_name $visaAddr "VOLT:PROT $p_vc(v_level)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
        puts $vi "CURR:LEV $p_vc(c_level);PROT:STAT ON"
        $gpib command_sent_wfile $device_name $visaAddr "CURR:LEV $p_vc(c_level);PROT:STAT ON"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
```

```
	#-----------------------------------------------------------------------------------------------------------------
	} err_code]} {
	    if {$err_code == $con_def_msg} {
	        $error_handle error_report_screen $visaAddr $c_err_msg
	    } else {
	        $error_handle error_report_screen $visaAddr "Unknown error"
	    }
	}
	#-----------------------------------------------------------------------------------------------------------------
	$error_handle execute_controlled_exit [$this error_report severity]
}

## Method to configure voltage protection (Voltage not modified) (Without changing output state)
# @param protect_v (parameter list) list with the following format:
# - \b Example: [list overv_level {3 V}]
#   + \b overv_level (string): Protection voltage level {<value> V|mV...MIN|MAX}
public method set_prot_psv { protect_v } {
	#-----------------------------------------------------------------------------------------------------------------
	if { [catch {
	#-----------------------------------------------------------------------------------------------------------------
	array set p_v $protect_v
	puts $vi "VOLT:PROT $p_v(overv_level)"
	$gpib command_sent_wfile $device_name $visaAddr "VOLT:PROT $p_v(overv_level)"
	$error_handle error_report_screen $visaAddr [$this error_report error_message]
	#-----------------------------------------------------------------------------------------------------------------
	} err_code]} {
	    if {$err_code == $con_def_msg} {
	        $error_handle error_report_screen $visaAddr $c_err_msg
	    } else {
	        $error_handle error_report_screen $visaAddr "Unknown error"
	    }
	}
	#-----------------------------------------------------------------------------------------------------------------
	$error_handle execute_controlled_exit [$this error_report severity]
}

## Method to configure current protection
# @param protect_c (parameter list) list with the following format:
# - \b Example: [list c_level {3 A}]
#   + \b c_level (string): Protection current level {<value> A|mA...MIN|MAX}
public method set_prot_psc { protect_c } {
	#-----------------------------------------------------------------------------------------------------------------
	if { [catch {
	#-----------------------------------------------------------------------------------------------------------------
	array set p_c $protect_c
	puts $vi "CURR:LEV $p_c(c_level);PROT:STAT ON"
	$gpib command_sent_wfile $device_name $visaAddr "CURR:LEV $p_c(c_level);PROT:STAT ON"
	$error_handle error_report_screen $visaAddr [$this error_report error_message]
	#-----------------------------------------------------------------------------------------------------------------
	} err_code]} {
	    if {$err_code == $con_def_msg} {
	        $error_handle error_report_screen $visaAddr $c_err_msg
	    } else {
	        $error_handle error_report_screen $visaAddr "Unknown error"
	    }
	}
	#-----------------------------------------------------------------------------------------------------------------
	$error_handle execute_controlled_exit [$this error_report severity]
}

## Method that Clears any OV (overvoltage), OC (overcurrent, unless set via external voltage control),OT (overtemperature), or
RI (remote inhibit) protection features.
# Warning: This function removes protection state and protection configuration
public method clear_prot {} {
	#-----------------------------------------------------------------------------------------------------------------
	if { [catch {
	#-----------------------------------------------------------------------------------------------------------------
	puts $vi "CURRENT:PROTECTION:STATE OFF"
	$gpib command_sent_wfile $device_name $visaAddr "CURRENT:PROTECTION:STATE OFF"
	$error_handle error_report_screen $visaAddr [$this error_report error_message]
	puts $vi "VOLT:PROT MAX"
	$gpib command_sent_wfile $device_name $visaAddr "VOLT:PROT MAX"
	$error_handle error_report_screen $visaAddr [$this error_report error_message]
	puts $vi "OUTP:PROT:CLE"
	$gpib command_sent_wfile $device_name $visaAddr "OUTP:PROT:CLE"
	$error_handle error_report_screen $visaAddr [$this error_report error_message]
	#-----------------------------------------------------------------------------------------------------------------
	} err_code]} {
	    if {$err_code == $con_def_msg} {
	        $error_handle error_report_screen $visaAddr $c_err_msg
	    } else {
	        $error_handle error_report_screen $visaAddr "Unknown error"
	    }
	}
	#-----------------------------------------------------------------------------------------------------------------
	$error_handle execute_controlled_exit [$this error_report severity]
}


## Method to read protection events produced: OV (Over Voltage): 1    OC (Over Current): 2    OT(Over Temperature): 16   RI
(Remote inhibit is active): 512     UNR (Power supply output is unregulated): 1024
# @return (integer): Event number
public method read_events {} {
	#-----------------------------------------------------------------------------------------------------------------
	if { [catch {
	#-----------------------------------------------------------------------------------------------------------------
	puts $vi "STAT:QUES?"
	set id [gets $vi]
	$gpib command_sent_wfile $device_name $visaAddr "STAT:QUES?"
	$error_handle error_report_screen $visaAddr [$this error_report error_message]

	#-----------------------------------------------------------------------------------------------------------------
	} err_code]} {
	    if {$err_code == $con_def_msg} {
	        $error_handle error_report_screen $visaAddr $c_err_msg
	    } else {
	            $error_handle error_report_screen $visaAddr "Unknown error"
	    }
	}
	#-----------------------------------------------------------------------------------------------------------------
	$error_handle execute_controlled_exit [$this error_report severity]

	if { $err_code == "" } {
	#-----------------------------------------------------------------------------------------------------------------
	return $id
	#-----------------------------------------------------------------------------------------------------------------
	}

}

## Method to read power supply outputs {voltage (V) | current (A) | power (W) }
# @param type (string): "voltage" (Reads voltage)  "current" (Reads current)     "power" (Reads power)   "all" (Reads voltage
```

```
        current power)
        # @return (float): Value of a single measure
        # @return (list): List with the value of all measurements {voltage | current | power}
        public method read_outputs { type } {
                #----------------------------------------------------------------------------------------------------
                if { [catch {
                #----------------------------------------------------------------------------------------------------
                    puts $vi "MEAS:VOLT?"
                    set voltage_read [gets $vi]

                    $gpib command_sent_wfile $device_name $visaAddr "MEAS:VOLT?"
                    $error_handle error_report_screen $visaAddr [$this error_report error_message]

                    puts $vi "MEAS:CURR?"
                    set current_read [gets $vi]

                    $gpib command_sent_wfile $device_name $visaAddr "MEAS:CURR?"
                    $error_handle error_report_screen $visaAddr [$this error_report error_message]

                    set power_read [expr $voltage_read*$current_read]
                    set reads_list [list current $current_read voltage $voltage_read power $power_read all {$voltage_read $current_read
                    $power_read}]
                    array set reads $reads_list

                #----------------------------------------------------------------------------------------------------
                } err_code]} {
                    if {$err_code == $con_def_msg} {
                        $error_handle error_report_screen $visaAddr $c_err_msg
                    } else {
                            $error_handle error_report_screen $visaAddr "Unknown error"
                    }
                }
                #----------------------------------------------------------------------------------------------------
                $error_handle execute_controlled_exit [$this error_report severity]

                if { $err_code == "" } {
                #----------------------------------------------------------------------------------------------------
                return $reads($type)
                #----------------------------------------------------------------------------------------------------
                }
         }
        ## \private
        # Method that setup the instrument in an error_handle object previusly declared
        # No parameters needed
        public method setup_error_handle {} {
            set possition [string last "::" $this]
            set dev_name [string range $this  [expr $possition+2] end]

            $error_handle add_instrument $dev_name $this
        }
        ## \private
        ## Method that returns the errors message or severty produced during the sequence
        # @param option string with the output option (error | severity)
        # @return (parameter_list) list with all the errors or max severity produced
        # - \b Example (error): {-101  "Sintax error"} {+0 "No error"}
        #   + \b First value: Error Code
        #   + \b Second value: Error message
        # - \b Example (severity): +5
        private method error_report { option } {
                #----------------------------------------------------------------------------------------------------
                if { [catch {
                #----------------------------------------------------------------------------------------------------
                set err_dict_output [list]
                set err_sev_list [list]


                puts $vi "SYST:ERR?"
                    set err [gets $vi]

                set val [expr [string first ","  $err 0]]
                set error_val [string range $err 0 [expr ($val-1)]]


                if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                        lappend err_dict_output "$err"
                        lappend err_sev_list 1
                } else {
                    set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                    lappend err_dict_output "$error_val,  \"$definition\""
                    lappend err_sev_list $severity_val
                }

                for {set x 0} { $err != "+0,\"No error\"" } {incr x} {
                    puts $vi "SYST:ERR?"
                    set err [gets $vi]

                    set val [expr [string first ","  $err 0]]
                    set error_val [string range $err 0 [expr ($val-1)]]



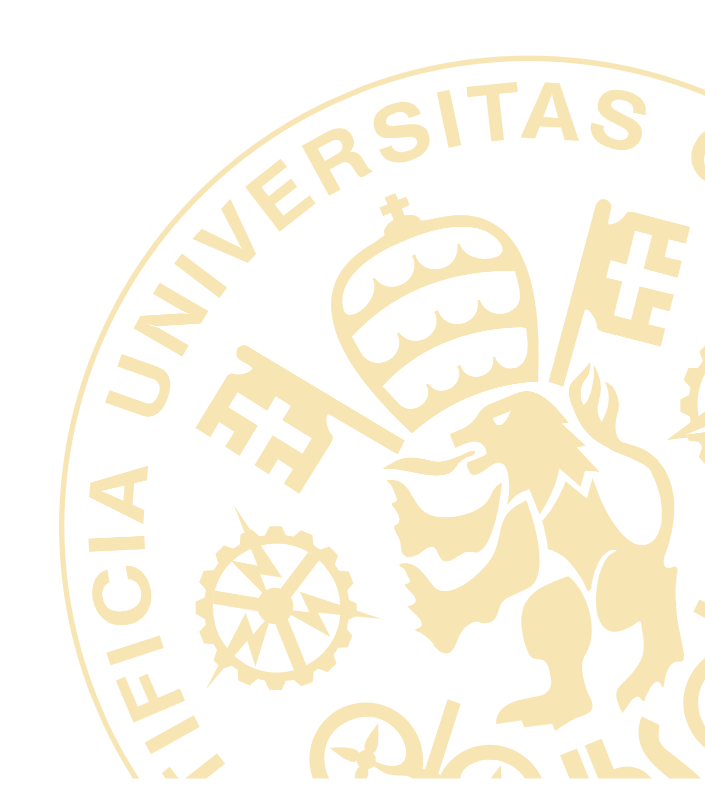
                    if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                        lappend err_dict_output "$err"
                        lappend err_sev_list 1
                    } else {
                        set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                        lappend err_dict_output "$error_val,  \"$definition\""
                        lappend err_sev_list $severity_val
                    }
                }
                foreach n $err_sev_list {
                    if {$n > $max} {
                        set max $n
                    }
                }
                set error_message $err_dict_output
                set severity $max
                #----------------------------------------------------------------------------------------------------
                } err_code]} {
                    if {$err_code == $con_def_msg} {
                        $error_handle  execute_controlled_exit $severity_nu
                    } else {
                            $error_handle error_report_screen $visaAddr "Unknown error"
                    }
                } else {
                #----------------------------------------------------------------------------------------------------
                    if { $severity == 0 && $option == "severity"} {
                    } else {
                        if {$severity != 0 && $option == "severity"} {
                            set max 0
                        }
                        eval return $$option
                    }
                }
                }
        }
    namespace export driver_gpib_hp6653a
}
```

Dicionario de errores

```
{
"+0":   {"No Error"                                                              :"0"},
"-100": {"Command error (generic command error)"                                 :"0"},
"-101": {"Invalid character"                                                     :"0"},
"-102": {"Syntax error (unrecognized command or data type)"                      :"0"},
"-103": {"Invalid separator (illegal character encountered in place of separator)" :"0"},
"-104": {"Data type error (e.g., “numeric or string expected, got block date”)"  :"0"},
"-105": {"GET not allowed ( <GET> inside a program message)"                     :"0"},
"-108": {"Parameter not allowed (too many parameters)"                           :"0"},
"-109": {"Missing parameter (too few parameters)"                                :"0"},
"-112": {"Program mnemonic too long (maximum 12 characters)"                     :"0"},
"-113": {"Undefined header (syntactical correct but not defined for this device)" :"0"},
"-121": {"Invalid character in number (e.g. alpha in decimal data, etc.)"        :"0"},
"-123": {"Exponent too large ( numeric overflow; exponent magnitude >32000)"     :"0"},
"-124": {"Too many digits (number too long; more than 255 digits received)"      :"0"},
"-128": {"Numeric data not allowed (numeric data not accepted where positioned)" :"4"},
"-131": {"Invalid suffix (unrecognized suffix, or suffix not appropriate)"       :"1"},
"-138": {"Suffix not allowed (numeric element does not allow suffixes)"          :"0"},
"-141": {"Invalid character data (bad character, or unrecognized)"               :"0"},
"-144": {"Character data too long (maximum length is 12 characters)"             :"0"},
"-148": {"Character data not allowed (character data not accepted where positioned)" :"0"},
"-150": {"String data error (generic string error)"                              :"0"},
"-151": {"Invalid string data (e.g., END received before close quote)"           :"0"},
"-158": {"String data not allowed (string data not accepted where positioned)"   :"0"},
"-160": {"Block data error (generic data block error)"                           :"0"},
"-161": {"Invalid block data (e.g., END received before length satisfied)"       :"0"},
"-168": {"Block data not allowed (block data not accepted where positioned)"     :"0"},
"-220": {"Parameter error"                                                       :"0"},
"-221": {"Settings conflict (uncoupled parameters)"                              :"0"},
"-222": {"Data out of range (e.g., outside the range of this device)"            :"1"},
"-223": {"Too much data (out of memory; block, string, or expression too long)"  :"0"},
"-240": {"Hardware error (device-dependent)"                                     :"0"},
"-241": {"Hardware missing (device-dependent)"                                   :"0"},
"-310": {"System error (device-dependent)"                                       :"0"},
"-313": {"Calibration memory lost (out of calibration due to memory failure)"    :"0"},
"-330": {"Self-test failed (more specific data after “;”)"                       :"0"},
"-350": {"Queue overflow (errors lost due to too many errors in queue)"          :"0"},
"-400": {"Query error (generic query error)"                                     :"0"},
"-410": {"Query INTERRUPTED (query followed by DAB or GET before response complete)" :"0"},
"-420": {"Query UNTERMINATED (addressed to talk, incomplete programming message received)" :"0"},
"-430": {"Query DEADLOCKED (too many queries in command string)"                 :"0"},
"-440": {"Query UNTERMINATED (query received after query for indefinite response)" :"0"}
}
```

PARTE V

DRIVER_GPIB_HP34401A



Librería de la clase

```
package provide driver_gpib_hp34401a 1.0
package require class_multimeter
package require Itcl

namespace eval driver_gpib_hp34401a {

      ##Namespace that implement a concrete multimeter hp34401a (inherit from multimeter class)
      namespace import ::itcl::*

      ::itcl::class driver_gpib_hp34401a {
        # Abstract class
        namespace import ::class_multimeter::*

        inherit ::class_multimeter::multimeter

        #Variables
        private variable visaAddr
        private variable on_off
        private variable vi
        private variable rc
        private variable rm
        private variable device_name
        private variable err
        private variable gpib
        private variable dict_err
        private variable err_dict_output
        private variable c_err_msg
        private variable connection_error_value
        private variable con_def_msg
        private variable con_def_msg_er
        private variable severity_nu
        private variable error_handle
        private variable max

        ## Constructor of the class
        #  @param _visa addrs (string) : Initialize the VISA device
        constructor {_visa_addrs _gpib d_error_handle} {
            #Pointer to $error_handle class
            set error_handle $d_error_handle

            #Pointer to driver_log class
            set gpib $_gpib

            # open device
            set visaAddr $_visa_addrs
            # get handle to default resource manager
            if { [catch { set rm [visa::open-default-rm] } rc] } {
                puts stderr "Error opening default resource manager\n$rc"
            } else {
                set rm [visa::open-default-rm]
            }

            # check if devide is opened
            if { [catch { set vi [visa::open $rm $visaAddr] } rc] } {
                puts "Error opening instrument `$visaAddr`\n$rc"
            } else {
                set vi [visa::open $rm "$visaAddr"]
                # Set proper timeout
                fconfigure $vi -timeout 500
            }

            # Get ID from instrument
            puts $vi "*IDN?"
            #Remove useless part of ID
            set _id [gets $vi]
            #Characters until second ","
            set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
            #Redefine device ID
            set id [string range $_id 0 $num]
            #Device name
            set device_name $id

            #Setup error dictionary
            set fp [open [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) setup sw source tcl
            driver_gpib_hp34401a driver_gpib_hp34401a_dict_error.json] r]
            set dict_err [read $fp]
            set dict_err [json::json2dict $dict_err]


            # SEVERITY CONNECTION ERROR VALUE
            set severity_nu 14
            set connection_error_value -1
            set c_err_msg [list "$connection_error_value, \"Connection error\" $severity_nu" "+0, \"No Error\" 0"]
            set con_def_msg "error writing \"$vi\": Unknown error"
            #Specific error report when disconection
            set c_err_msg_er "$connection_error_value, \"Connection error\" $severity_nu"

            #Add instrument in error_handle
            $this setup_error_handle

            #Initialize maximum severity
            set max 0


        }

        ## Turn ON/OFF display
        #  @param on_off (string) : "on" (Turns on the display)  "off" (Turns off the display)
        #
        #  Example: display on
        public method display { on_off } {
            #---------------------------------------------------------------------------------------------------
            if { [catch {
            #---------------------------------------------------------------------------------------------------
                    puts $vi "DISPlay $on_off"
                    $gpib command_sent_wfile $device_name $visaAddr "DISPlay $on_off"
                    $error_handle error_report_screen $visaAddr [$this error_report error_message]
            #---------------------------------------------------------------------------------------------------
            } err_code]} {

                # Report communication error

                if {$err_code == $con_def_msg} {
                    $error_handle error_report_screen $visaAddr $c_err_msg
                } else {
                    $error_handle error_report_screen $visaAddr "Unknown error"
                }
            }
            #---------------------------------------------------------------------------------------------------
            $error_handle  execute_controlled_exit [$this error_report severity]
        }

        ## Read resistance(x2)
```

```
# @param r_confx2 (parameter_list)  list with the following format: [ list  range { <value> MOHM|KOHM|OHM...|MIN|MAX|DEF}
resolution { <value> MOHM|KOHM|OHM...|MIN|MAX|DEF}]
#
# Example: read_resx2 [list range {5 ohm} resolution {1 ohm}]
public method read_resx2 { r_confx2} {
    #-----------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------

        array set r_paramx2 $r_confx2


        puts $vi "CONF:RES $r_paramx2(range), $r_paramx2(resolution)"
        $gpib command_sent_wfile $device_name $visaAddr "CONF:RES $r_paramx2(range), $r_paramx2(resolution)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
        puts $vi "READ?"
        set id [gets $vi]
        $gpib command_sent_wfile $device_name $visaAddr "READ?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-----------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #-----------------------------------------------------------------------------------------------------------
        return $id
    #-----------------------------------------------------------------------------------------------------------
    }

}

## Read resistance(x4)
# @param r_conf (parameter_list)  list with the following format: [ list  range { <value> MOHM|KOHM|OHM...|MIN|MAX|DEF}
resolution { <value> MOHM|KOHM|OHM...|MIN|MAX|DEF}]
#
# Example: read_resx4 [list range {5 ohm} resolution {1 ohm}]
public method read_resx4 { r_confx4} {
    #-----------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------

    array set r_paramx4 $r_confx4
    puts $vi "CONF:FRES $r_paramx4(range), $r_paramx4(resolution)"
    $gpib command_sent_wfile $device_name $visaAddr "CONF:FRES $r_paramx4(range), $r_paramx4(resolution)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    puts $vi "READ?"
    set id [gets $vi]
    $gpib command_sent_wfile $device_name $visaAddr "READ?"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-----------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]


    if { $err_code == "" } {
    #-----------------------------------------------------------------------------------------------------------
    return $id
    #-----------------------------------------------------------------------------------------------------------
    }

}

## Read voltage dc
# @param v_conf (parameter_list) list with the following format: [ list range { <value> kV|V|mV...|MIN|MAX|DEF}    resolution {
<value> kV|V|mV ...|MIN|MAX|DEF}]
#
# Example: read_voltage_dc [list range {5 mV} resolution {1 mV}]
public method read_voltage_dc { v_conf_dc } {
    #-----------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------
        array set v_param $v_conf_dc
        puts $vi "CONF:VOLT:DC $v_param(range), $v_param(resolution)"
        $gpib command_sent_wfile $device_name $visaAddr "CONF:VOLT:DC $v_param(range), $v_param(resolution)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
        puts $vi "READ?"
    set id [gets $vi]
        $gpib command_sent_wfile $device_name $visaAddr "READ?"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
    #-----------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]


    if { $err_code == "" } {
    #-----------------------------------------------------------------------------------------------------------
        return $id
    #-----------------------------------------------------------------------------------------------------------
    }


}

## Read voltage ac
# @param v_conf (parameter_list) list with the following format: [ list range { <value> kV|V|mV...|MIN|MAX|DEF}    resolution {
<value> kV|V|mV ...|MIN|MAX|DEF}]
#
# Example: read_voltage_ac [list range {5 V} resolution {1 mV}]
public method read_voltage_ac { v_conf_ac } {
    #-----------------------------------------------------------------------------------------------------------
```

```
    if { [catch {
    #------------------------------------------------------------------------------------------------
            array set v param $v conf ac
            puts $vi "CONF:VOLT:AC $v_param(range), $v_param(resolution)"
            $gpib command_sent_wfile $device_name $visaAddr "CONF:VOLT:AC $v_param(range), $v_param(resolution)"
            $error handle error report screen $visaAddr [$this error report error_message]
            puts $vi "READ?"
        set id [gets $vi]
        $gpib command sent wfile $device name $visaAddr "READ?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error handle error report screen $visaAddr $c err msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------
    $error handle  execute controlled exit [$this error report severity]


    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------
    return $id
    #------------------------------------------------------------------------------------------------
    }


}


## Read current dc
# @param c_conf (parameter_list) list with the following format: [ list range { <value> A|mA ...|MIN|MAX|DEF}    resolution {
<value> A|mA ...|MIN|MAX|DEF}]
#
# Example: read_current_dc [list range {200 mA} resolution {1 A}]
public method read_current_dc { c_conf_dc } {
    #------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------
            array set c_param $c_conf_dc
            puts $vi "CONF:CURR:DC $c_param(range), $c_param(resolution)"
            $gpib command_sent_wfile $device name $visaAddr "CONF:CURR:DC $c param(range), $c param(resolution)"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
            puts $vi "READ?"
        set id [gets $vi]
            $gpib command_sent_wfile $device_name $visaAddr "READ?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err code == $con def msg} {
            $error_handle error_report_screen $visaAddr $c_err_msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------
    $error_handle  execute_controlled_exit [$this error_report severity]


    if { $err_code == "" } {
    #------------------------------------------------------------------------------------------------
        return $id
    #------------------------------------------------------------------------------------------------
    }
}


## Read current ac
    # @param c_conf (parameter_list) list with the following format: [ list range { <value> A|mA ...|MIN|MAX|DEF}    resolution
    { <value> A|mA ...|MIN|MAX|DEF}]
    #
    # Example: read_current_ac [list range {200 mA} resolution {1 A}]
    public method read_current_ac { c_conf_ac } {
    #------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------
            array set c param $c conf ac
            puts $vi "CONF:CURR:AC $c_param(range), $c_param(resolution)"
            $gpib command_sent_wfile $device_name $visaAddr "CONF:CURR:AC $c_param(range), $c_param(resolution)"
            $error handle error report screen $visaAddr [$this error report error_message]
            puts $vi "READ?"
        set id [gets $vi]
            $gpib command sent wfile $device name $visaAddr "READ?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error handle error report screen $visaAddr $c err msg
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #------------------------------------------------------------------------------------------------
    $error handle  execute controlled exit [$this error report severity]

    if { $err code == "" } {
    #------------------------------------------------------------------------------------------------
        return $id
    #------------------------------------------------------------------------------------------------
    }

}


## Read frequency
# @param f conf (parameter list) list with the following format: [ list  range { <value> kHz|Hz ...|MIN|MAX|DEF}    resolution
{ <value> kHz|Hz ...|MIN|MAX|DEF}]
#
# Example: read freq [list range {20 kHz}  resolution {100 hz}]
public method read_freq { f_conf } {
    #------------------------------------------------------------------------------------------------
    if { [catch {
    #------------------------------------------------------------------------------------------------
        array set f_param $f_conf
        puts $vi "CONF:FREQ $f param(range),$f param(resolution)"
        $gpib command_sent_wfile $device_name $visaAddr "CONF:FREQ $f_param(range),$f_param(resolution)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
```

```
                puts $vi "READ?"
                $error_handle error_report_screen $visaAddr [$this error_report error_message]

                set id [gets $vi]

                $gpib command_sent_wfile $device_name $visaAddr "READ?"
                $error_handle error_report_screen $visaAddr [$this error_report error_message]

            #-------------------------------------------------------------------------------------------------------
            } err_code]} {
                if {$err_code == $con_def_msg} {
                    $error_handle error_report_screen $visaAddr $c_err_msg
                } else {
                    $error_handle error_report_screen $visaAddr "Unknown error"
                }
            }
            #-------------------------------------------------------------------------------------------------------
            $error_handle  execute_controlled_exit [$this error_report severity]

            if { $err_code == "" } {
            #-------------------------------------------------------------------------------------------------------
                return $id
            #-------------------------------------------------------------------------------------------------------
            }
        }

        ## Read current value
        #
        # Example: read
        public method read_meas {} {
            #-------------------------------------------------------------------------------------------------------
            if { [catch {
            #-------------------------------------------------------------------------------------------------------
                puts $vi "READ?"
                $error_handle error_report_screen $visaAddr [$this error_report error_message]
                set id [gets $vi]
                $gpib command_sent_wfile $device_name $visaAddr "READ?"
                $error_handle error_report_screen $visaAddr [$this error_report error_message]

            #-------------------------------------------------------------------------------------------------------
            } err_code]} {
                if {$err_code == $con_def_msg} {
                    $error_handle error_report_screen $visaAddr $c_err_msg
                } else {
                    $error_handle error_report_screen $visaAddr "Unknown error"
                }
            }
            #-------------------------------------------------------------------------------------------------------
            $error_handle  execute_controlled_exit [$this error_report severity]

            if { $err_code == "" } {
            #-------------------------------------------------------------------------------------------------------
            return $id
            #-------------------------------------------------------------------------------------------------------
            }

        }

        ## Auto adds instrument in error_handle
        public method setup_error_handle {} {
            set possition [string last "::" $this]
            set dev_name [string range $this  [expr $possition+2] end]

            $error_handle add_instrument $dev_name $this
        }

        ## Return error list from the error queue
        #  @return err (parameter_list) list with all the errors present in the error queue
        #
        #  Example_: error_report
        private method error_report { option } {
            #-------------------------------------------------------------------------------------------------------
            if { [catch {
            #-------------------------------------------------------------------------------------------------------
            set err_dict_output [list]
            set err_sev_list [list]


                puts $vi "SYST:ERR?"
            set err [gets $vi]

            set val [expr [string first ","  $err 0]]
            set error_val [string range $err 0 [expr ($val-1)]]


            if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                lappend err_dict_output "$err"
                lappend err_sev_list 1
            } else {
                set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                lappend err_dict_output "$error_val,  \"$definition\""
                lappend err_sev_list $severity_val
            }

                for {set x 0} { $err != "+0,\"No error\"" } {incr x} {
                puts $vi "SYST:ERR?"
                set err [gets $vi]

                set val [expr [string first ","  $err 0]]
                set error_val [string range $err 0 [expr ($val-1)]]



                if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                    lappend err_dict_output "$err"
                    lappend err_sev_list 1
                } else {
                    set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                    lappend err_dict_output "$error_val,  \"$definition\""
                    lappend err_sev_list $severity_val

                }
              }

            foreach n $err_sev_list {
                if {$n > $max} {
                    set max $n
                }
            }

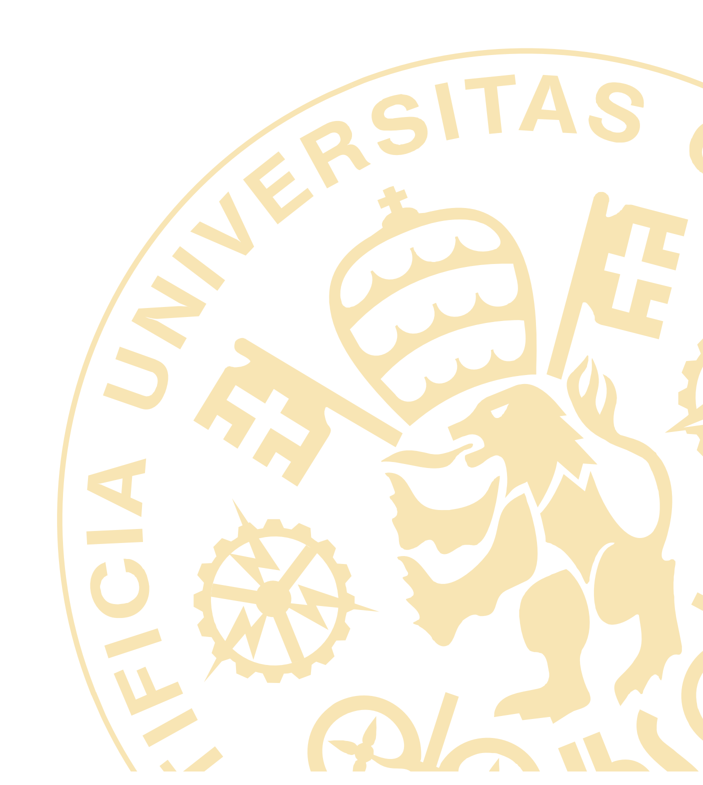
            set error_message $err_dict_output
            set severity $max
            #-------------------------------------------------------------------------------------------------------
            } err_code]} {
                if {$err_code == $con_def_msg} {
                    $error_handle  execute_controlled_exit $severity_nu
                } else {
                    $error_handle error_report_screen $visaAddr "Unknown error"
                }
            } else {
            #-------------------------------------------------------------------------------------------------------
                if { $severity == 0 && $option == "severity"} {
                } else {
                    if {$severity != 0 && $option == "severity"} {
                        set max 0
                    }
                    eval return $$option
                }
            }
        }
    }
    namespace export driver_gpib_hp34401a
}
```

Dicionario de errores

```
{
"+0":   {"No error"                          :"0"},
"-101": {"Invalid character"                 :"0"},
"-102": {"Syntax error"                      :"0"},
"-103": {"Invalid separator"                 :"0"},
"-104": {"Data type error"                   :"0"},
"-105": {"GET not allowed"                   :"0"},
"-108": {"Parameter not allowed"             :"0"},
"-109": {"Missing parameter"                 :"0"},
"-112": {"Program mnemonic too long"         :"0"},
"-113": {"Undefined header"                  :"0"},
"-121": {"Invalid character in number"       :"0"},
"-123": {"Numeric overflow"                  :"0"},
"-124": {"Too many digits"                   :"0"},
"-131": {"Invalid suffix"                    :"0"},
"-138": {"Suffix not allowed"                :"0"},
"-148": {"Character data not allowed"        :"0"},
"-158": {"String data not allowed"           :"0"},
"-160": {"Block data errors"                 :"0"},
"-161": {"Block data errors"                 :"0"},
"-162": {"Block data errors"                 :"0"},
"-163": {"Block data errors"                 :"0"},
"-164": {"Block data errors"                 :"0"},
"-165": {"Block data errors"                 :"0"},
"-166": {"Block data errors"                 :"0"},
"-167": {"Block data errors"                 :"0"},
"-168": {"Block data errors"                 :"0"},
"-170": {"Expression errors"                 :"0"},
"-171": {"Expression errors"                 :"0"},
"-172": {"Expression errors"                 :"0"},
"-173": {"Expression errors"                 :"0"},
"-174": {"Expression errors"                 :"0"},
"-175": {"Expression errors"                 :"0"},
"-176": {"Expression errors"                 :"0"},
"-176": {"Expression errors"                 :"0"},
"-176": {"Expression errors"                 :"0"},
"-211": {"Trigger ignored"                   :"0"},
"-213": {"Init ignored"                      :"0"},
"-214": {"Trigger deadlock"                  :"0"},
"-221": {"Settings conflict"                 :"0"},
"-222": {"Data out of range"                 :"0"},
"-223": {"Too much data"                     :"0"},
"-224": {"Illegal parameter value"           :"0"},
"-230": {"Data stale"                        :"0"},
"-330": {"Self-test failed"                  :"0"},
"-350": {"Too many errors"                   :"0"}
}
```

PARTE VI

DRIVER_GPIB_KIKUPLZ150U



Librería de la clase

```
package provide driver_gpib_kikuplz150u 1.0
package require class_electronic_load
package require Itcl

namespace eval driver_gpib_kikuplz150u {

    namespace import ::itcl::*

    ::itcl::class driver_gpib_kikuplz150u {
        # Abstract class
        namespace import ::class_electronic_load::*
          inherit ::class_electronic_load::electronic_load

        #Variables
        private variable visaAddr
        private variable on_off
        private variable vi
        private variable rc
        private variable rm
        private variable device_name
        private variable err
        private variable mode
        private variable dict_err
        private variable gpib
        private variable err_dict_output
        private variable c_err_msg
        private variable connection_error_value
        private variable con_def_msg
        private variable con_def_msg_er
        private variable severity_nu
        private variable error_handle
        private variable max


         ## Constructor of the class
         #  @param _visa_addrs (string) : Initialize the VISA device
        constructor {_visa_addrs _gpib d_error_handle} {
            #Pointer to $error_handle class
            set error_handle $d_error_handle

            #Pointer to driver_gpib
            set gpib $_gpib

            # open device
            set visaAddr $_visa_addrs
            set visaAddr "GPIB0::1::INSTR"
            # get handle to default resource manager
            if { [catch { set rm [visa::open-default-rm] } rc] } {
                puts stderr "Error opening default resource manager\n$rc"
            } else {
                set rm [visa::open-default-rm]
            }

            # check if devide opened
            if { [catch { set vi [visa::open $rm $visaAddr] } rc] } {
                puts "Error opening instrument `$visaAddr`\n$rc"
            } else {
                set vi [visa::open $rm "$visaAddr"]
                # Set proper timeout
                fconfigure $vi -timeout 500
            }

            # Get ID from instrument
            puts $vi "*IDN?"
            #Remove useless part of ID
            set _id [gets $vi]
            #Characters until second ","
            set num [expr [string first "," $_id  [expr [string first "," $_id]+1]]-1]
            #Redefine device ID
            set id [string range $_id 0 $num]
            #Device name
            set device_name $id

            #Setup error dictionary
            set fp [open [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) setup sw source tcl
            driver_gpib_kikuplz150u driver_gpib_kikuplz150u_dict_error.json] r]
            set dict_err [read $fp]
            set dict_err [json::json2dict $dict_err]

            # SEVERITY CONNECTION ERROR VALUE
            set severity_nu 14
            set connection_error_value -1
            set c_err_msg [list "$connection_error_value, \"Connection error\" $severity_nu" "0, \"No Error\" 0"]
            set con_def_msg "error writing \"$vi\": Unknown error"

            #Specific error report when disconection
            set c_err_msg_er "$connection_error_value, \"Connection error\" $severity_nu"

            #Add instrument in error handle
            $this setup_error_handle

            #Initialize maximum severity
            set max 0

         }

        ## Set working mode of the dl: CC|CR|CV|CCCV|CRCV
        # CC: Constant current      CR: Constant resistance mode    CV: Constant voltage mode       CCCV: Constant current mode +
        constant voltage mode
        # @param l_dl_mode (parameter_list) list with the following format: [ list channel {CH1, CH2, CH3... ALL | NONE}    mode
        {CC|CR|CV|CCCV|CRCV}]
        #
        # Example: set_mode [list channel {ch1 ch4} mode {cc}]
         public method set_mode { l_dl_mode} {
            #-----------------------------------------------------------------------------------------------------
            if { [catch {
            #-----------------------------------------------------------------------------------------------------

            array set dl_mode $l_dl_mode

            #Select more than one device
            foreach chan $dl_mode(channel) {
                set l_channel([incr i]) $chan
                lappend l_channelcoup "$chan,"
            }
            puts $vi "INST:COUP $l_channelcoup"
            $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]

            foreach chan $dl_mode(channel) {
                puts $vi "INST $chan"
                #Reporting
```

```
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    puts $vi "FUNCtion $dl_mode(mode)"
    $gpib command_sent_wfile $device_name $visaAddr "FUNCtion $dl_mode(mode)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
 }


## Turn ON | OFF device input
# @param state (string) with the value: on | off
#
# Example: input on
 public method input {state} {
    #-------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-------------------------------------------------------------------------------------------------------------

    puts $vi "INP $state"
    $gpib command_sent_wfile $device_name $visaAddr "INP $state"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]


    #-------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-------------------------------------------------------------------------------------------------------------
    # $error_handle execute_controlled_exit [$this error_report severity]

}

## Turn ON | OFF device output
# @param state (string) with the value: on | off
#
# Example: output off
public method output {state} {
    #-------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-------------------------------------------------------------------------------------------------------------

    puts $vi "OUTP $state"
    $gpib command_sent_wfile $device_name $visaAddr "OUTP $state"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

## Set device protection of current, voltage and power
# @param l_dl_protection (parameter list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE} current
{<value> mA|A... MIN|MAX|no}} voltage {value mV|V... MIN|MAX|no} power {value W... MIN|MAX|no}]
#
# Example: set_protection [list channel {ch1 ch2} current {1 A} voltage {2 V} power {2 W}]
 public method set_protection { l_dl_protection } {
    #-------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-------------------------------------------------------------------------------------------------------------

    array set dl_protection $l_dl_protection

    #Select more than one device
    foreach chan $dl_protection(channel) {
        set l_channel([incr i]) $chan
        append l_channelcoup "$chan,"
    }
    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    foreach chan $dl_protection(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    if($dl_protection(current) != "no") {
    #Set current protection
        puts $vi "CURR:PROT $dl_protection(current)"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "CURR:PROT $dl_protection(current)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #Enable current protection
        puts $vi "CURR:PROT:ACT"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr puts $vi "CURR:PROT:ACT"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    if($dl_protection(voltage) != "no") {
    #Set voltage protection
        puts $vi "VOLT:PROT:UND $dl_protection(voltage)"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr puts $vi "VOLT:PROT:UND $dl_protection(voltage)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #Enable / Disable voltage protection:
```

```
        puts $vi "VOLT:PROT:STAT ON"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr puts $vi "VOLT:PROT:STAT ON"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    if($dl_protection(power) != "no") {
    #Set pwer protection
        puts $vi "POW:PROT $dl_protection(power)"

        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr puts $vi "POW:PROT $dl_protection(power)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #Enable / Disable pwer protection
        puts $vi "POW:PROT:STAT on"
        $gpib command_sent_wfile $device_name $visaAddr puts $vi "POW:PROT:STAT on"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    #-----------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

## Disable all protections and clears protection flags
#
# Example disable_prot
public method disable_prot {} {
    #-----------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------------

    puts $vi "CURR:PROT MAX"
    $gpib command_sent_wfile $device_name $visaAddr "CURR:PROT MAX"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    puts $vi "VOLT:PROT:UNDER MAX"
    $gpib command_sent_wfile $device_name $visaAddr "VOLT:PROT:UNDER MAX"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    puts $vi "INP:PROT:CLE"
    $gpib command_sent_wfile $device_name $visaAddr "INP:PROT:CLE"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    puts $vi "OUTP:PROT:CLE"
    $gpib command_sent_wfile $device_name $visaAddr "OUTP:PROT:CLE"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-----------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]

}

## Set conductance value
# @param l_dl_conductance (parameter_list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE}
conductance {value sie|msie} mode {AUTO | LOW | MED | HIGH}]
#
# Example: set_conductance [list channel {ch1 ch2} conductance {0.2 sie} mode auto]
public method set_conductance { l_dl_conductance} {
    #-----------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------------

    array set dl_conductance $l_dl_conductance

    #Select more than one device
    foreach chan $dl_conductance(channel) {
        set l_channel([incr i]) $chan
        lappend l_channelcoup "$chan,"
    }
    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    foreach chan $dl_conductance(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    #Unit default
    set cond(2) "sie"
    #Units used
    foreach unit $dl_conductance(conductance) {
        set cond([incr a]) $unit
    }

    #Automatic Conductance Module selector:
    if {$dl_conductance(mode) == "auto" || $dl_conductance(mode) == "AUTO"} {
        set cond_value $cond(1)
        if {$cond(2) == "msie" || $cond(2) == "MSIE"} {
            set cond_value [expr $cond(1)/1000]
        }

        if {$cond_value <= 0.2 } {
            puts $vi "COND:RANG LOW"
            $gpib command_sent_wfile $device_name $visaAddr  "COND:RANG LOW"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]

        } elseif {$cond_value <= 2 || $cond_value > 0.2} {
            puts $vi "COND:RANG MED"
            $gpib command_sent_wfile $device_name $visaAddr  "COND:RANG MED"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]

        } elseif {$cond_value > 2} {
            puts $vi "COND:RANG HIGH"
```

```
            $gpib command_sent_wfile $device_name $visaAddr  "COND:RANG HIGH"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        }
    } else {
        puts $vi "COND:RANG $dl_conductance(mode)"
        $gpib command_sent_wfile $device_name $visaAddr "COND:RANG $dl_conductance(mode)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }


    #set conductance
    puts $vi "COND $dl_conductance(conductance)"
    $gpib command_sent_wfile $device_name $visaAddr "COND $dl_conductance(conductance)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-----------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

## Set current value
# @param mode (parameter_list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE} current {value
mA|A... MIN|MAX} mode {AUTO | LOW | MED | HIGH}]
public method set_current {l_dl_current} {
    #-----------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------
    array set dl_current $l_dl_current

    foreach chan $dl_current(channel) {
        set l_channel([incr i]) $chan
        lappend l_channelcoup "$chan,"
    }
    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    foreach chan $dl_current(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    #Unit default
    set curr(2) "A"

    #Units used
    foreach unit $dl_current(current) {
        set curr([incr a]) $unit
    }

    #Automatic Current Module selector:
    if {$dl_current(mode) == "auto" || $dl_current(mode) == "AUTO"} {
        set curr_value $curr(1)
        if {$curr(2) == "ma" || $curr(2) == "MA"} {
            set curr_value [expr $curr(1)/1000]
        }

        if {$curr_value <= 0.315 } {
            puts $vi "CURR:RANG LOW"
            $gpib command_sent_wfile $device_name $visaAddr  "CURR:RANG LOW"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]

        } elseif {$curr_value <= 3.15 || $curr_value < 0.315} {
            puts $vi "CURR:RANG MED"
            $gpib command_sent_wfile $device_name $visaAddr  "CURR:RANG MED"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]

        } elseif {$curr_value > 3.15} {
            puts $vi "CURR:RANG HIGH"
            $gpib command_sent_wfile $device_name $visaAddr  "CURR:RANG HIGH"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        }
    } else {
        puts $vi "CURR:RANG $dl_current(mode)"
        $gpib command_sent_wfile $device_name $visaAddr   "CURR:RANG $dl_current(mode)"
    }

    #set current
    puts $vi "CURR $dl_current(current)"
    $gpib command_sent_wfile $device_name $visaAddr  "CURR $dl_current(current)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-----------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
 }

## Set voltage value
# @param l_dl_voltage (parameter_list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE} voltage
{value mV|V... MIN|MAX} mode {AUTO | LOW | HIGH}]
#
# Example: set_voltage [list channel {ch1} voltage {5 V} mode auto]
public method set_voltage {l_dl_voltage} {
    #-----------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------
    array set dl_voltage $l_dl_voltage

    #Select more than one device
    foreach chan $dl_voltage(channel) {
        set l_channel([incr i]) $chan
        lappend l_channelcoup "$chan,"
    }
    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]
```

```
    foreach chan $dl_voltage(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }


    #Unit default
    set volt(2) "V"
    #Units used
    foreach unit $dl_voltage(voltage) {
        set volt([incr a]) $unit
    }

    #Automatic voltage Module selector:
    if {$dl_voltage(mode) == "auto" || $dl_voltage(mode) == "AUTO"} {
        set volt_value $volt(1)
        if {$volt(2) == "mv" || $volt(2) == "MV"} {
            set volt_value [expr $volt(1)/1000]
        }

        if {$volt_value <= 15.75 } {
            puts $vi "VOLT:RANG LOW"
            $gpib command_sent_wfile $device_name $visaAddr  "VOLT:RANG MED"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } elseif {$volt_value > 15.75} {
            puts $vi "VOLT:RANG HIGH"
            $gpib command_sent_wfile $device_name $visaAddr  "VOLT:RANG HIGH"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        }

    } else {
        puts $vi "VOLT:RANG $dl_voltage(mode)"
        $gpib command_sent_wfile $device_name $visaAddr  "VOLT:RANG $dl_voltage(mode)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    puts $vi "VOLT $dl_voltage(voltage)"
    $gpib command_sent_wfile $device_name $visaAddr  "VOLT $dl_voltage(voltage)"age]
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

## Set soft start
# @param l_soft (parameter_list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE} sst {value mS|S...
MIN|MAX}]
#
# Example: soft_start [list channel {ch1 ch5} sst {5 ms}]
public method soft_start {l_soft} {
    #-------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-------------------------------------------------------------------------------------------------------------
    array set soft $l_soft
    #Select more than one device
    foreach chan $soft(channel) {
        set l_channel([incr i]) $chan
        lappend l_channelcoup "$chan,"
    }
    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    foreach chan $soft(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }


    puts $vi "FUNC:SST $soft(sst)"
    $gpib command_sent_wfile $device_name $visaAddr "FUNC:SST $soft(sst)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

# Set current slope per micro Sec
# @param l_slew (parameter_list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE} slew {value} mode
{AUTO | LOW | MED | HIGH}]
#
# Example: set_slew [list channel {ch1 ch5} slew 0.5 mode auto]
public method set_slew {l_slew} {
    #-------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-------------------------------------------------------------------------------------------------------------
    array set slew $l_slew
    #Select more than one device
    foreach chan $slew(channel) {
        set l_channel([incr i]) $chan
        lappend l_channelcoup "$chan,"
    }
    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    foreach chan $slew(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $l_channel($i)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }
```

```
    puts $vi "CURR:SLEW $slew(slew)"
    $gpib command_sent_wfile $device_name $visaAddr "CURR:SLEW $slew(slew)"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    #-----------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

# Set load level as a pulse or as a pulse train
# @param l_load_tran (parameter list) list with the following format: [ list channel {CH1 CH2 CH3... ALL | NONE} max {value
mA|A... MIN|MAX} min {value mA|A... MIN|MAX} slew {value} mode {{train freq_value} | single | stop}]
#
# Example: set_load_tran [list channel {ch1 ch2} max {1 A} min {0.5 A} slew 0.01 mode {train 5}]
# Example: set_load_tran [list channel {ch1 ch2} mode stop]
public method set_pulse_train {l_load_tran} {
    #-----------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------

    array set load_tran $l_load_tran

    #Select more than one device
    foreach chan $load_tran(channel) {
            set l_channel([incr i]) $chan
            lappend l_channelcoup "$chan,"
    }

    puts $vi "INST:COUP $l_channelcoup"
    $gpib command_sent_wfile $device_name $visaAddr "INST:COUP $l_channelcoup"
    $error_handle error_report_screen $visaAddr [$this error_report error_message]

    foreach chan $load_tran(channel) {
        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    #Stop transition
    if { $load_tran(mode) == "stop"} {
        puts $vi "PULS OFF"

    } else {

        puts $vi "CURR:SLEW $load_tran(slew)"
        $gpib command_sent_wfile $device_name $visaAddr "CURR:SLEW $load_tran(slew)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        set mode(1) 0
        set mode(2) 1
        set i 0

        foreach type $load_tran(mode) {
            set mode([incr i]) $type
        }

        #Set Voltage
        $this set_current [list channel "$load_tran(channel)" current "$load_tran(max)" mode "auto"]
        #Set Pulse
        puts $vi "PULS:LEV:CURR $load_tran(min)"
        $gpib command_sent_wfile $device_name $visaAddr "PULS:LEV:CURR $load_tran(min)"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]


        if {$mode(1)=="train"} {
            puts $vi "PULS:FREQ $mode(2)"
            $gpib command_sent_wfile $device_name $visaAddr "PULS:FREQ $mode(2)"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
            puts $vi "PULS ON"
            $gpib command_sent_wfile $device_name $visaAddr "PULS ON"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]


        } elseif {$mode(1)=="single"} {
            puts $vi "PULS OFF"

            $gpib command_sent_wfile $device_name $visaAddr "PULS OFF"
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
            $this set_current [list channel "$load_tran(channel)" current "$load_tran(min)" mode "auto"]
            $this set_current [list channel "$load_tran(channel)" current "$load_tran(max)" mode "auto"]

        }
    }

    #-----------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]
}

# # Measure current
# @param channel (parameter_list) list with the following format: [ list {CH1 CH2 CH3... ALL | NONE}]
#
# Example: read_current {ch1 ch2}
public method read_current {channel} {
    #-----------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------
    set c_meas [list]
    #Select more than one device
    foreach chan $channel {

        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        puts $vi "MEAS:CURR:DC?"
        lappend c_meas [gets $vi]
```

```
        $gpib command_sent_wfile $device_name $visaAddr "MEAS:CURR:DC?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }

    #-----------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #-----------------------------------------------------------------------------------------------------------------
    return $c_meas
    #-----------------------------------------------------------------------------------------------------------------
    }
}

# # Measure voltage
# @param channel (parameter_list) list with the following format: [ list {CH1 CH2 CH3... ALL | NONE}]
#
# Example: read_voltage {ch1 ch2}
public method read_voltage {channel} {
    #-----------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------------
    set v_meas [list]
    foreach chan $channel {

        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        puts $vi "MEAS:VOLT:DC?"
        lappend v_meas [gets $vi]
        $gpib command_sent_wfile $device_name $visaAddr "MEAS:VOLT:DC?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

    }

    #-----------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #-----------------------------------------------------------------------------------------------------------------
    return $v_meas
    #-----------------------------------------------------------------------------------------------------------------
    }
}

# # Measure power
# @param channel (parameter_list) list with the following format: [ list {CH1 CH2 CH3... ALL | NONE}]
#
# Example: read_power {ch1 ch2}
public method read_power {channel} {
    #-----------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------------
    set p_meas [list]
    #Select more than one device
    foreach chan $channel {

        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        puts $vi "MEAS:POW:DC?"
        lappend p_meas [gets $vi]
        $gpib command_sent_wfile $device_name $visaAddr "MEAS:POW:DC?"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]
    }
    #-----------------------------------------------------------------------------------------------------------------
    } err_code]} {
        if {$err_code == $con_def_msg} {
            $error_handle error_report_screen $visaAddr [$this error_report error_message]
        } else {
            $error_handle error_report_screen $visaAddr "Unknown error"
        }
    }
    #-----------------------------------------------------------------------------------------------------------------
    $error_handle execute_controlled_exit [$this error_report severity]

    if { $err_code == "" } {
    #-----------------------------------------------------------------------------------------------------------------
    return $p_meas
    #-----------------------------------------------------------------------------------------------------------------
    }
}


# # Return conductance configured
# @param channel (parameter_list) list with the following format: [ list {CH1 CH2 CH3... ALL | NONE}]
#
# Example: read_conductance {ch1 ch2}
public method read_conductance {channel} {
    #-----------------------------------------------------------------------------------------------------------------
    if { [catch {
    #-----------------------------------------------------------------------------------------------------------------
    set p_meas [list]
    #Select more than one device
    foreach chan $channel {

        puts $vi "INST $chan"
        #Reporting
        $gpib command_sent_wfile $device_name $visaAddr "INST $chan"
        $error_handle error_report_screen $visaAddr [$this error_report error_message]

        puts $vi "COND?"
        lappend p_meas [gets $vi]
```

```
                $gpib command_sent_wfile $device_name $visaAddr "COND?"
                $error_handle error_report_screen $visaAddr [$this error_report error_message]
            }
            #-----------------------------------------------------------------------------------------------------------
            } err_code]} {
                if {$err_code == $con_def_msg} {
                    $error_handle error_report_screen $visaAddr [$this error_report error_message]
                } else {
                    $error_handle error_report_screen $visaAddr "Unknown error"
                }
            }
            #-----------------------------------------------------------------------------------------------------------
            $error_handle execute_controlled_exit [$this error_report severity]

            if { $err_code == "" } {
            #-----------------------------------------------------------------------------------------------------------
            return $p_meas
            #-----------------------------------------------------------------------------------------------------------
            }
        }




        ## Auto adds instrument in error_handle
        public method setup_error_handle {} {
            set possition [string last "::" $this]
            set dev_name [string range $this  [expr $possition+2] end]

            $error_handle add_instrument $dev_name $this
        }

        ## Return error list from the error queue
        #  @return err (parameter_list) list with all the errors present in the error queue
        #
        #  Example_: error_report
        private method error_report { option } {
            #-----------------------------------------------------------------------------------------------------------
            if { [catch {
            #-----------------------------------------------------------------------------------------------------------
            set err_dict_output [list]
            set err_sev_list [list]


            puts $vi "SYST:ERR?"
            set err [gets $vi]

            set val [expr [string first ","  $err 0]]
            set error_val [string range $err 0 [expr ($val-1)]]


            if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                    lappend err_dict_output "$err"
                    lappend err_sev_list 1
            } else {
                set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                lappend err_dict_output "$error_val,  \"$definition\""
                lappend err_sev_list $severity_val
            }

            for {set x 0} { $err != "0,\"No error\"" } {incr x} {
                puts $vi "SYST:ERR?"
                set err [gets $vi]

                set val [expr [string first ","  $err 0]]
                set error_val [string range $err 0 [expr ($val-1)]]



                if { [catch { set definition [lindex [dict get $dict_err $error_val] 0 ]} rc ]} {
                    lappend err_dict_output "$err"
                    lappend err_sev_list 1
                } else {
                    set severity_val [lindex [dict get $dict_err $error_val] 1 ]
                    lappend err_dict_output "$error_val,  \"$definition\""
                    lappend err_sev_list $severity_val

                }
            }



            foreach n $err_sev_list {
                if {$n > $max} {
                    set max $n
                }
            }

            set error_message $err_dict_output
            set severity $max

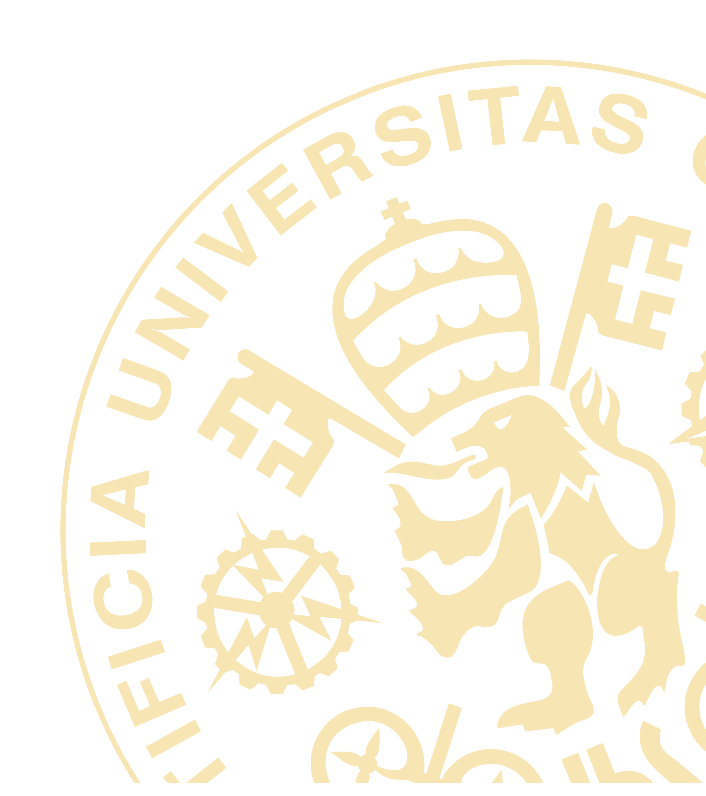
            #-----------------------------------------------------------------------------------------------------------
            } err_code]} {
                if {$err_code == $con_def_msg} {
                    $error_handle execute_controlled_exit $severity_nu
                } else {
                    $error_handle error_report_screen $visaAddr "Unknown error"
                }
            } else {
            #-----------------------------------------------------------------------------------------------------------
                if { $severity == 0 && $option == "severity"} {
                } else {
                    if {$severity != 0 && $option == "severity"} {
                        set max 0
                    }
                    eval return $$option
                }
            }
        }
    }
    namespace export driver_gpib_kikuplz150u
}
```

Dicionario de errores

```
{
"0":    {"No error"                                              : "0"},
"21":   {"Operation denied due to ALARM state"                  : "0"},
"22":   {"Operation denied due to PROGRAM running"              : "0"},
"23":   {"Operation denied due to SWITCH running"               : "0"},
"24":   {"Operation denied due to INPUT ON"                     : "0"},
"27":   {"Operation denied due to incompatible FUNCTION MODE"   : "0"},
"31":   {"Operation denied due to incompatible PROGRAM MODE"    : "0"},
"-100": {"Command error"                                         : "0"},
"-101": {"Invalid character"                                     : "0"},
"-102": {"Syntax error"                                          : "0"},
"-103": {"Invalid separator"                                     : "0"},
"-104": {"Data type error"                                       : "0"},
"-105": {"GET not allowed"                                       : "0"},
"-108": {"Parameter not allowed"                                 : "0"},
"-109": {"Missing parameter"                                     : "0"},
"-110": {"Command header error"                                  : "0"},
"-120": {"Numeric data error"                                    : "0"},
"-130": {"Suffix error"                                          : "0"},
"-131": {"Invalid suffix"                                        : "0"},
"-134": {"Suffix too long"                                       : "0"},
"-138": {"Suffix not allowed"                                    : "0"},
"-140": {"Character data error"                                  : "0"},
"-150": {"String data error"                                     : "0"},
"-160": {"Block data error"                                      : "0"},
"-170": {"Expression error"                                      : "0"},
"-180": {"Macro error"                                           : "0"},
"-200": {"Execution error (generic)"                             : "0"},
"-300": {"Device-specific error(generic)"                        : "0"},
"-350": {"Queue overflow"                                        : "0"},
"-400": {"Query error (generic)"                                 : "0"}
}
```

PARTE VII

DRIVER_RS232_TEKTTPS2024



Librería de la clase

```
package provide driver_rs232_tekttps2024 1.0
#  class abstract
# package require driver_class_scanner

package require Itcl

namespace eval driver_rs232_tekttps2024 {
    namespace import ::itcl::*

        ::itcl::class driver_rs232_tekttps2024 {
            private variable vi
            private variable fp
            private variable config_dir

            ## Constructor of the class
                #  @param serial_port (string) : Initialize RS232 communication
            constructor { serial_port _config_dir } {
                set vi [open $serial_port: RDWR]

            #   set vi [open COM1: RDWR]

                fconfigure $vi -blocking 1
                fconfigure $vi -buffering full
                fconfigure $vi -encoding binary
                fconfigure $vi -mode 19200,n,8,1
                fconfigure $vi -translation binary
                fconfigure $vi -eofchar {}
                fconfigure $vi -timeout 10000

            #   Configuration files location
                set config_dir $_config_dir
            }

            ## Destructor of the class
            # Closes the channel
            destructor {
                close $vi
            }


            ## Get configuration from osciloscope
                #  @param file_name (string) : name of the file in which the configuration is going to be saved
                #
                #  Example: getconfig conf1

            public method getconfig { file_name } {

                set log_add [file join [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) work tektronix
                $file_name.SET]]
                set fp [open $log_add w]

                set settings [list]

                lappend settings "HEADER?"
                lappend settings "DATA?"
                lappend settings "DISPLAY?"
                lappend settings "ACQUIRE?"
                lappend settings "CH1?"
                lappend settings "CH2?"
                lappend settings "CH3?"
                lappend settings "CH4?"
                lappend settings "HORIZONTAL?"
                lappend settings "TRIGGER?"
                lappend settings "SELECT?"
                lappend settings "CURSOR?"
                lappend settings "MEASUREMENT?"
                lappend settings "CURSOR?"
                lappend settings "MATH?"


                foreach conf $settings {
                    puts $vi $conf
                    flush $vi

                    set conf_out ""
                    while {$conf_out == ""} {
                        set conf_out [gets $vi]
                    }

                    puts $fp $conf_out
                }

                close $fp

            }


            ## Setup osciloscope configuration from file
                #  @param file_name (string) : name of the file that contains the configuration
                #
                #  Example: setconfig conf1

            public method   setconfig { file_name } {

                set log_add [file join [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK)/setup/sw/source/tcl/
                $config_dir/$file_name.SET]]
                set fp [open $log_add r]


                set config [gets $fp]

                while { $config != "" } {
                    puts $vi $config
                    flush $vi
                    set config [gets $fp]
                    after 2000
                }
            }

            ## Creates an image of the screen of the osciloscope
            #  @param bmp_name (string) : name of the bmp file that will be created
            #
            # Example: hard_copy imagen1

            public method hard_copy { bmp_name } {

                #Check if osciloscope is busy or not

                puts $vi "BUSY?"
                flush $vi
                set flag [gets $vi]

                if { $flag == ":BUSY 0" || $flag == "0" } {
```

```
        set log_add [file join [file join $::env(HOME_PROJECTS)/ $::env(NAME_PROJECT) $::env(USERWORK) work tektronix
        $bmp_name.bmp]]
        set fp [open $log_add w]
        # File configuration
        fconfigure $fp -encoding binary
        fconfigure $fp -translation binary

        # Image format settings
        puts $vi "HARDCopy:FORMat BMP"
        flush $vi

        # Start scann
        puts $vi "HARDcopy start"
        flush $vi

        # Read number of bytes in the buffer queue
        set data [lindex [fconfigure $vi -queue] 0]

        # Wait until data receiving
        while {$data == 0} {
           set data [lindex [fconfigure $vi -queue] 0]
        }

        # Wait for header

        while { $data < 54 } {
           set data [lindex [fconfigure $vi -queue] 0]
        }

        # Read header
        set header [read $vi 54]
        # Read image size in header
        set size_bin [string range $header 2 5]
        binary scan $size_bin i size

        # Write header in image file
        puts -nonewline $fp $header

        # Initialize counter value with the header length
        set counter 54

        # Read data until reaching the total image size
        while { $counter < $size} {
           set data [lindex [fconfigure $vi -queue] 0]

            if { $data >= 1 } {
                set counter [expr $counter+$data]
                puts -nonewline $fp [read $vi $data]
            }
        }

        close $fp

        return 0

    } else {
        return 1
    }


}




## Returns wave signal in values & time scale of values
#
#  @param wave_settings (parameter_list) list with the following format:  [list mode {auto | manual | time} channel { 1 | 2
| 3 | 4} ]
#   If "time" mode is choosen, channel value is not needed.
#   "time" option just returns the time scale of the graphic
#
# Example: wave_curve [list mode time]
# Example: wave_curve [list mode auto channel 1]
public method wave_curve { wave_settings } {
    array set settings $wave_settings

    if { $settings(mode) == "auto" } {
        $this auto_acquisisiton
        set wave_form [$this waveform $settings(channel)]
        return $wave_form

    } elseif { $settings(mode) == "manual" } {
        $this manual_acquisition
        set wave_form [$this waveform $settings(channel)]
        return $wave_form

    } elseif { $settings(mode) == "time" } {
        # Scale of time cursor
        puts $vi "HORizontal:MAIn:SCALe?"
        flush $vi
        set time_scl_string [gets $vi]
        set time_scl [string range $time_scl_string 23 end]
        return time_scl
    }

}

## Configure acquisition time intervals in auto mode [FULL LENGHT]
private method auto_acquisisiton {} {
    # Wave adquisition settings
    puts $vi "DATa:STARt 1"
    puts $vi "DATa:STOP 2500"
    flush $vi

}

## Configure acquisition time intervals in manual mode [USE CURSORS TO DELIMIT TIME INTERVAL]
private method manual_acquisition {} {

    # Horizontal time offset of cursor
    puts $vi "HORizontal:MAIn:POSition?"
    flush $vi
    set h_gt [gets $vi]
    set h_globaltime [string range $h_gt 26 end]

    # Scale of time cursor
    puts $vi "HORizontal:MAIn:SCALe?"
    flush $vi
    set time_scl [gets $vi]
    set time_scale [string range $time_scl 23 end]
```

```
            # Cursor 1 position
            puts $vi "CURSor:VBArs:POSITION1?"
            flush $vi
            set pos_c1 [gets $vi]
            set position_cursor1 [string range $pos_c1 24 end]

            # Cursor 2 position
            puts $vi "CURSor:VBArs:POSITION2?"
            flush $vi
            set pos_c2 [gets $vi]
            set position_cursor2 [string range $pos_c2 24 end]

            # Data Start calculation (Cursor 1)
            set t_start [expr floor((($position_cursor1-$h_globaltime)+5*$time_scale)/(10*$time_scale)*2500)]
            if {$t_start <= 0 } {
                set t_start 1
            }

            # Data Stop calculation (Cursor 2)
            set t_stop [expr floor((($position_cursor2-$h_globaltime)+5*$time_scale)/(10*$time_scale)*2500)]
            if {$t_stop <= 0 } {
                set t_stop 1
            }

            # Wave adquisition settings
            puts $vi "DATa:STARt $t_start"
            puts $vi "DATa:STOP $t_stop"
            flush $vi
        }


        ## Returns wave signal in values
        #
        #  @param channel (string) : channel number whose signal will be scanned  { 1 | 2 | 3 | 4}
        #
        # Example: waveform 1
        private method waveform {channel} {
            # Wave acquisition settings
            puts $vi "DATa:ENCdg RIBinary"
            puts $vi "DATa:WIDth 1"
            puts $vi "DATa:SOUrce CH$channel"
            flush $vi

            # Start scan
            puts $vi "CURVE?"
            flush $vi

            # Header:
            set header [read $vi 9]
            set num_length_bytes [string range $header 8 end]
            set wave_data_length [read $vi $num_length_bytes]
            # Wave Data:
            set wave_data_bin [read $vi [expr $wave_data_length+1]]
            # Conversion from binary to ASCII
            binary scan [string range $wave_data_bin 0 end-2] c* wave_data_ascii

            #====== Wave form offset and configuration ======

            # Escalas de voltage
            puts $vi "CH$channel:SCAle?"
            flush $vi
            set v_scale_str [gets $vi]
            set v_scale [string range $v_scale_str 11 end]

            # Cursor Vertical possition [screen squares]
            puts $vi "CH$channel:POSition?"
            flush $vi
            set v_possition_str [gets $vi]
            set v_possition [string range $v_possition_str 14 end]


            #=========== Wave form output ============
            set counter 0
            set new_wave_data [list]
            foreach value $wave_data_ascii {
                set wave_data_processed [expr ($value/25.0)*$v_scale - $v_possition* $v_scale ]
                set rounded_number [expr {double(round(1000*$wave_data_processed))/1000}]
                lappend new_wave_data $rounded_number
            }

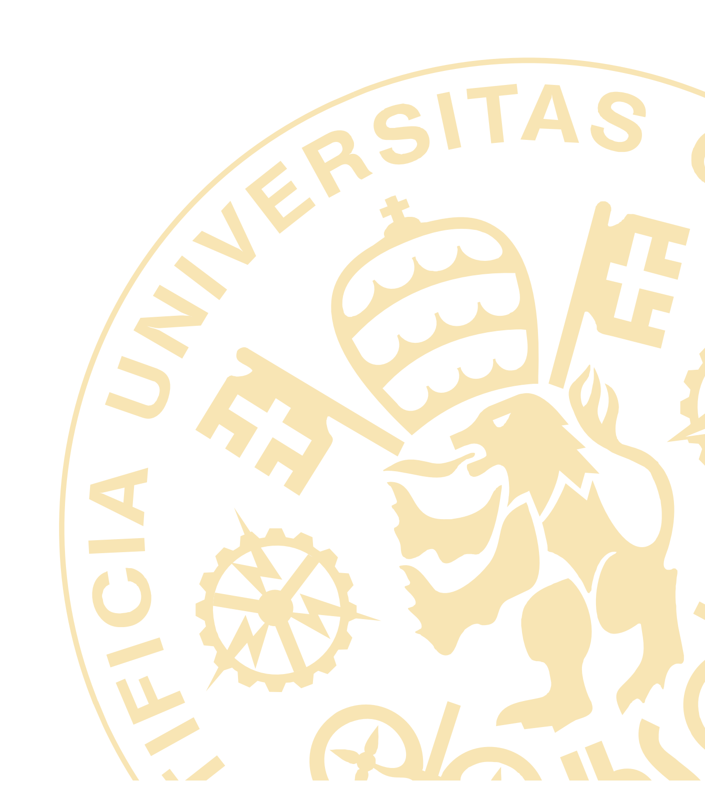
            return $new_wave_data
        }

    }
    namespace export driver_rs232_tekttps2024

}
```

PARTE VIII

CLASES ABSTRACTAS



Clase Escáner

```
## \file
#  File that contains the Scanner abstract library
package provide class_scanner 1.0

## Namespace with the abstract class of the Scanner
namespace eval class_scanner {

    ## This class is the abstract class with the common functions of a Scanner
    ::itcl::class scanner {

        ## Method to switch on | off the screen of the instrument.
        # @param mode string with the format: on  | off
        public method display { mode } {}

      ## Method to read DC Voltage
        # @param parameter_list list with the following format:
        # - \b Example: [list channel {105:110,215} range {5 mV}  resolution {1 mV}]
        #   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
        #   + \b range  (string): Range of the measure
        #   + \b resolution (string): Resolution of the measure
        # @return measures (list) list with all reads
        public method read_voltage_dc { parameter_list} {}

        ## Method to read AC Voltage
        # @param parameter_list list with the following format:
        # - \b Example: [list channel {105:110,215} range {5 mV}  resolution {1 mV}]
        #   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
        #   + \b range  (string): Range of the measure
        #   + \b resolution (string): Resolution of the measure
        # @return measures (list) list with all reads
        public method read_voltage_ac  { parameter_list} {}

        ## Method to read DC current (Read user manual to see which channels are compatible with current measures)
        # @param parameter_list list with the following format:
        # - \b Example: [list channel {105:110,215} range {200 mA} resolution {1 mA}]
        #   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
        #   + \b range  (string): Range of the measure
        #   + \b resolution (string): Resolution of the measure
        # @return measures (list) list with all reads
        public method read_current_dc { parameter_list } {}

        ## Method to read AC current (Read user manual to see which channels are compatible with current measures)
        # @param parameter_list list with the following format:
        # - \b Example: [list channel {105:110,215} range {200 mA} resolution {1 mA}]
        #   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
        #   + \b range  (string): Range of the measure
        #   + \b resolution (string): Resolution of the measure
        # @return measures (list) list with all reads
        public method read_current_ac { parameter_list } {}

        ## Method to read 2 wires resistances
        # @param parameter_list list with the following format:
        # - \b Example: [list channel {105:110,215} range {200 mohm} resolution {1 mohm}]
        #   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
        #   + \b range  (string): Range of the measure
        #   + \b resolution (string): Resolution of the measure
        # @return measures (list) list with all reads
        public method read_resistancex2 { parameter_list} {}

        ## Method to read 4 wires resistances
        # @param parameter_list list with the following format:
        # - \b Example: [list channel {105:110,215} range {200 mohm} resolution {1 mohm}]
        #   + \b channel (string): Indicates the channels that will be read (:) [From - to] || (,) single
        #   + \b range  (string): Range of the measure
        #   + \b resolution (string): Resolution of the measure
        # @return measures (list) list with all reads
        public method read_resistancex4 { parameter_list} {}

        ## \private
          ## Method that returns the errors message or severty produced during the sequence
          # @param option string with the output option (error | severity)
          # @return (parameter_list) list with all the errors or max severity produced
          private method error_report {option} {}

          ## \private
          # Method that setup the instrument in an error_handle object previusly declared
          # No parameters needed
        private method setup_error_handle {} {}


   }

    # export namespaces
    namespace export scanner
}
```

Clase Fuente de alimentación

```
##\file
# File that contains the Power Supply abstract library

package provide class_powersupply 1.0

## Namespace with the abstract class of the Power Supply
namespace eval class_powersupply {

   ## This class is the abstract class with the common functions of a Power Supply
   ::itcl::class powersupply {

      ## Method to switch on | off the screen of the instrument.
      # @param mode string with the format: on  | off
      public method display { mode } {}

      ## Method to switch the state of the output (current | voltage) of the instrument.
      # @param state (string): on | off
      public method output { state } {}

      ## Method to configure voltage and current level without changing the output state.
      # @param output_vc (list) with the following format:
      # - \b Example: [list voltage {50 mV} current {3 mA}]
      #   + \b voltage: Voltage level
      #   + \b current: Current level
      public method set_psvc { output_vc } {}

      ## Method to configure the voltage level without changing the output state.
      # @param output_v (string) with the following format:
      # - \b Example: {30 V}
      public method set_psv { output_v } {}

      ## Method to configure the current maximun level without changing the output state.
      # @param output_c (string) with the following format:
      # - \b Example: {2 A}
      public method set_psc { output_c } {}

      ## Method to configure the voltage level, the voltage protection level and the current protection level without changing the
      output state.
      # @param parameter_list (list) with the following format:
      # - \b Example: [list v_level {70 mV}   overv_level {1 V}   c_level {60 mA}]
      #   + \b v_level: Voltage output level
      #   + \b overv_level: Voltage protection level
      #   + \b c_level: Current protection level
      public method set_prot_psvc { parameter_list } {}

      ## Method to configure just the voltage protection level.
      # @param protect_v (string) with the following format:
      # - \b Example: {25 V}
      public method set_prot_psv { protect_v } {}

      ## Method to configure just the current protection level.
      # @param protect_c (string) with the following format:
      # - \b Example: {1 A}
      public method set_prot_psc { protect_c } {}

      ## Method that clears any OV (overvoltage), OC (overcurrent, unless set via external voltage control),OT (overtemperature), or RI
      (remote inhibit) protection features.
      # No parameters needed
      public method cle_protect {} {}

      ## Method that reads any event produced due to a protection issue. (Over Voltage)     (Over Current)  OT(Over Temperature) RI
      (Remote inhibit is active)    UNR (Power supply output is unregulated)
      # @return event (integer): Value with the event produced
      public method read_events {} {}

      ## Method that returns the output value indicated: (Current | Voltage | Power)
      # @param type (string) with the following format:
      # - \b Example: current
      # - \b Example: voltage
      # - \b Example: power
      # - \b Example: all
      # @return measure (value | list value) with the measures indicated
      public method read_outputs { type } {}

      ## \private
      ## Method that returns the errors message or severty produced during the sequence
      # @param option string with the output option (error | severity)
      # @return (parameter_list) list with all the errors or max severity produced
       private method error_report {} {}

      ## \private
      # Method that setup the instrument in an error_handle object previously declared
      # No parameters needed
       private method setup_error_handle {} {}

   }

   # export namespaces
   namespace export powersupply
}
```

Clase Multímetro

```
##\file
# File that contains the Multimeter abstract library

package provide class_multimeter 1.0

## Namespace with the abstract class of the Multimeter
namespace eval class_multimeter {

   ## This class is the abstract class with the common functions of a Multimeter
   ::itcl::class multimeter {

    ## Method to switch on | off the screen of the instrument
    # @param mode string with the following format:
    # - \b Example: on  | off
    public method display { mode } {}

    ## Method to read a 2 wire resistance
    # @param parameter list list with the following format:
    # - \b Example: [list range {5 ohm} resolution {1 ohm}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
    public method read_resx2 { parameter_list} {}

    ## Method to read a 4 wire resistance
    # @param parameter_list list with the following format:
    # - \b Example: [list range {5 ohm} resolution {1 ohm}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
    public method read_resx4 { parameter_list} {}

    ## Method to read DC Voltage
    # @param parameter_list list with the following format:
    # - \b Example: [list range {5 mV} resolution {1 mV}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
    public method read_voltage_dc { parameter_list  } {}

    ## Method to read AC Voltage
    # @param parameter list list with the following format:
    # - \b Example: [list range {5 V} resolution {1 V}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
      public method read_voltage_ac { parameter_list } {}

    ## Method to read DC Current
    # @param parameter_list list with the following format:
    # - \b Example: [list range {5 A} resolution {1 mA}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
    public method read_current_dc { parameter_list } {}

    ## Method to read AC Voltage
    # @param parameter_list list with the following format:
    # - \b Example: [list range {5 A} resolution {1 mA}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
      public method read_current_ac { parameter_list } {}

    ## Method to read the frequency of the signal
    # @param parameter list list with the following format:
    # - \b Example: [list range {20 kHz}  resolution {100 hz}]
    #   + \b range  (string): Range of the measure
    #   + \b resolution (string): Resolution of the measure
    # @return measure (value): value measured
    public method read_freq { parameter_list } {}

    ## Method to repeat the last measure configured previously
    # No params recquired
    # @return measure (value): value measured
    public method read_meas {} {}

    ## \private
    ## Method that returns the errors message or severty produced during the sequence
    # @param option string with the output option (error | severity)
    # @return (parameter list) list with all the errors or max severity produced
      private method error_report {option} {}

    ## \private
    # Method that setup the instrument in an error_handle object previusly declared
    # No parameters needed
      private method setup_error_handle {} {}

  }

  # export namespaces
   namespace export multimeter
}
```

Clase Carga Dinámica

```
##\file
# File that contains the Electronic Load abstract library

package provide class_electronic_load 1.0

## Namespace with the abstract class of the Electronic Load
namespace eval class_electronic_load {

   ## This class is the abstract class with the common functions of an Electronic Load
   ::itcl::class electronic_load {

    ## Method to set the mode of the Electonic Load: CC|CR|CV|CCCV|CRCV
    # CC: Constant current      CR: Constant resistance mode    CV: Constant voltage mode       CCCV: Constant current mode + constant
    voltage mode
    # @param parameter list list with the following format:
    # - \b Example: [list channel CH1   mode CRCV]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b mode (string): Mode in which the selected channel will work.
    public method set_mode { parameter_list } {}

    ## Method to switch the state of the input (current | voltage) of the instrument
    # @param state (string): on | off
    public method input {state} {}

    ## Method to switch the state of the output (current | voltage) of the instrument
    # @param state (string): on | off
    public method output {state} {}

    ## Method to set the protection levels of the instrument
    # @param parameter list list with the following format:
    # - \b Example: [list channel {ch1 ch2} current {1 A} voltage {2 V} power {2 W}]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b current (string): Current protection value
    #   + \b voltage (string): Voltage protection value
    #   + \b power (string): Power protection value
    public method set_protection { parameter_list } {}

    ## Method to disable all protections and clear protection flags
    # No parameters needed
    public method disable_prot {} {}

    ## Method to set the conductance level when mode CR | CRCV is activated
    # @param parameter_list list with the following format:
    # - \b Example: [list channel {ch1 ch2} conductance {0.2 sie} mode auto]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b conductance (string): Conductance value
    #   + \b mode (string): Set the range of the mode (AUTO | LOW | MED | HIGH)
    public method set_conductance { parameter_list} {}

    ## Method to set the current level when mode CC | CCCV is activated
    # @param parameter list list with the following format:
    # - \b Example: [ list channel {ch1 ch2} current {1 mA} mode low]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b current (string): Current value
    #   + \b mode (string): Set the range of the mode (AUTO | LOW | MED | HIGH)
    public method set_current {parameter_list} {}

    ## Method to set the voltage level when mode CV | CCCV | CRCV is activated
    # @param parameter_list list with the following format:
    # - \b Example: [list channel {ch1} voltage {5 V} mode auto]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b voltage (string): Voltage value
    #   + \b mode (string): Set the range of the mode (AUTO | LOW | MED | HIGH)
    public method set_voltage {parameter_list} {}

    ## Method to set the soft start time
    # @param parameter_list list with the following format:
    # - \b Example: [list channel {ch1 ch5} sst {5 ms}]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b sst (string): soft start time
    public method soft_start {parameter_list} {}

    ## Method to set the current slope per micro Sec
    # @param parameter_list list with the following format:
    # - \b Example: [list channel {ch1 ch5} slew 0.5 mode auto]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b slew (value): Current slope (A) per micro Sec
    #   + \b mode (string): Set the range of the mode (AUTO | LOW | MED | HIGH)
    public method set_slew {parameter_list} {}

    ## Method to set current as a single pulse | pulse train
    # @param parameter_list list with the following format:
    # - \b Example: [list channel {ch1 ch2} max {1 A} min {0.5 A} slew 0.01 mode {train 5}]
    # - \b Example: [list channel {ch1 ch2} max {1 A} min {0.5 A} slew 0.01 mode {single}]
    # - \b Example: [list channel {ch1 ch2} mode stop]
    #   + \b channel (string): Channel of the instrument that will be commanded
    #   + \b max (string): Maximun current value per pulse
    #   + \b min (string): Minimun current value per pulse
    #   + \b slew (value): Current slope (A) per micro Sec
    #   + \b mode (string): Select working mode (train | single | stop)
    #   + \b train (value): Oscilation fecquency
    public method et_pulse_train {parameter_list} {}

    ## Method to read the conductance programmed
    # @param parameter_list list with the following format:
    # - \b Example: {ch1 ch2}
    # @return string with the value measured
    public method read_conductance {parameter_list} {}

    ## \private
    ## Method that returns the errors message or severty produced during the sequence
    # @param option string with the output option (error | severity)
    # @return (parameter_list) list with all the errors or max severity produced
    private method error_report {option} {}

    ## \private
    # Method that setup the instrument in an error_handle object previusly declared
    # No parameters needed
    private method setup_error_handle {} {}


   }

   # export namespaces
   namespace export electronic_load
}
```

DOCUMENTO IV

ESTUDIO DE COSTES



Índice general

Capítulo 1

Recursos

Esta parte del presupuesto recogerá el número recursos que participan en la composición y construcción del sistema de test.

1. Componentes del sistema

1.1. Dispositivos de medición

Dispositivos de medición	Modelo	Unidades
Fuente de alimentación	HP 6653A	1
Multímetro	HP 34401a	1
Escaner	Agilent 34970a	1
Módulo carga dinámica	Kikusui PLZ 150u	5
Osciloscopio	Tektronix 2024	1

Tabla 1. Recursos correspondientes a los dispositivos de medición

1.2. Dispositivos de control

Dispositivos de control	Modelo	Unidades
Ordenador	-	1

Tabla 2. Recursos correspondientes a los dispositivos de control

1.3. Otros

Otros	Unidades
USB/GPIB Interface High-Speed USB 2.0	1
Cable GPIB (2 m)	3
Cable RS232 de 9 pines D-Sub	1
Carcasa para Carga Dinámica KRB3-PLZ-50F	1

Tabla 3. Recursos correspondientes a otros elementos del sistema

2. Software del sistema

2.1. Software requerido

Software	Unidades	Horas de uso
GPIB Comunication Software	1	320
Tcl 8.6.1 (ActiveTcl)	1	360

Tabla 4. Recursos correspondientes al software utilizado

2.2. Diseño de librerías

Diseño de Librerías	Horas de trabajo
Driver_GPIB	20
Error_handle	60
Driver_gpib_agil34970a	60
Driver_gpib_hp6653a	60
Driver_gpib_hp34401a	60
Driver_gpib_kikuplz150u	40
Driver_rs232_tekttps2024	40
Secuencia de test: Verificación Convertidor	20
HORAS TOTALES:	360

Tabla 5. Recursos correspondientes al diseño de las librerías del sistema

3. Formación del usuario

3.1. Manuales y Cursos

Aprendizaje y formación	Unidades
Manual de usuario TCL 8.6	1
Curso de introducción a TCL (4h)	1

Tabla 6. Recursos correspondientes al material formativo para futuros usuarios

Capítulo 2

Precios unitarios

Esta parte del presupuesto recogerá el precio de cada uno de los recursos que participan en la composición y construcción del sistema de test.

1. Componentes del sistema

1.1. Dispositivos de medición

Dispositivos de medición	Modelo	Precio/Unidad
Fuente de alimentación	HP 6653A	2.833,00 €
Multímetro	HP 34401a	850,00 €
Escaner	Agilent 34970a	1.280,00 €
Módulo carga dinámica	Kikusui PLZ 150u	4.470,00 €
Osciloscopio	Tektronix 2024	3.316,00 €

Tabla 7. Precios unitarios correspondientes a los dispositivos de medición

1.2. Dispositivos de control

Dispositivos de control	Modelo	Precio/Unidad
Ordenador	-	550,00 €

Tabla 8. Precios unitarios correspondientes a los dispositivos de control

1.3. Otros

Otros	Precio/Unidad
USB/GPIB Interface High-Speed USB 2.0	406 €
Cable GPIB (2 m)	68,22 €
Cable RS232 de 9 pines D-Sub	9 €
Carcasa para Carga Dinámica KRB3-PLZ-50F	47 €

Tabla 9. Precios unitarios correspondientes a otros elementos del sistema

2. Software del sistema

2.1. Software requerido

Software	Coste por licencia
GPIB Comunication Software	Libre distribución
Tcl 8.6.1 (ActiveTcl)	Libre distribución

Tabla 10. Precios unitarios correspondientes al software utilizado

2.2. Diseño de librerías

Diseño de librerías	Coste (€/hora)
Driver_GPIB	40,00 €
Error_handle	60,00 €
Driver_gpib_agil34970a	50,00 €
Driver_gpib_hp6653a	50,00 €
Driver_gpib_hp34401a	50,00 €
Driver_gpib_kikuplz150u	50,00 €
Driver_rs232_tekttps2024	70,00 €
Secuencia de test: Verificación Convertidor	60,00 €

Tabla 11. Precios unitarios correspondientes al diseño de las librerías del sistema

3. Formación del usuario

3.1. Manuales y Cursos

Aprendizaje y formación	Coste
Manual de usuario TCL 8.6	50,95 €
Curso de introducción a TCL (4h)	89 €

Tabla 12. Precios unitarios correspondientes al material formativo para futuros usuarios

Capítulo 3

Sumas parciales

Sumando los precios unitarios de cada recurso por su cantidad, se obtiene el precio unitario de cada recurso:

1. Componentes del sistema

1.1. Dispositivos de medición

Dispositivos de medición	Modelo	Precio / Ud.	Unidades	Coste total
Fuente de alimentación	HP 6653A	2.833 €	1	2.833 €
Multímetro	HP 34401a	850 €	1	850 €
Escaner	Agilent 34970a	1.280 €	1	1.280 €
Módulo carga dinámica	Kikusui PLZ 150u	894 €	5	4.470 €
Osciloscopio	Tektronix 2024	3.316 €	1	3.316 €
			TOTAL:	12.749 €

Tabla 13. Sumas parciales correspondientes a los dispositivos de medición

1.2. Dispositivos de control

Dispositivos de control	Modelo	Precio / Ud.	Unidades	Coste total
Ordenador	-	550 €	1	550 €
			TOTAL:	550 €

Tabla 14. Sumas parciales correspondientes a los dispositivos de control

1.3. Otros

Otros	Precio / Ud.	Unidades	Coste total
USB/GPIB Interface High-Speed USB 2.0	406 €	1	406 €
Cable GPIB (2 m)	68,22 €	3	205 €
Cable RS232 de 9 pines D-Sub	9 €	1	9 €
Carcasa para Carga Dinámica KRB3-PLZ-50F	47 €	1	47 €
		TOTAL:	667 €

Tabla 15. Sumas parciales correspondientes a otros elementos del sistema

2. Software del sistema

2.1. Software requerido

Software	Unidades	Coste por licencia
GPIB Comunication Software	1	Libre distribución
Tcl 8.6.1 (ActiveTcl)	1	Libre distribución
	TOTAL:	0 €

Tabla 16. Sumas parciales correspondientes al software utilizado

2.2. Diseño de librerías

Diseño de librerías	Horas	Precio €/h	Coste Total
Driver_GPIB	20	40,00 €	800,00 €
Error_handle	60	60,00 €	3.600,00 €
Driver_gpib_agil34970a	60	50,00 €	3.000,00 €
Driver_gpib_hp6653a	60	50,00 €	3.000,00 €
Driver_gpib_hp34401a	60	50,00 €	3.000,00 €
Driver_gpib_kikuplz150u	40	50,00 €	2.000,00 €
Driver_rs232_tekttps2024	40	70,00 €	2.800,00 €
Secuencia de test: Verificación Convertidor	20	60,00 €	1.200,00 €
		TOTAL:	19.400,00 €

Tabla 17. Sumas parciales correspondientes al diseño de las librerías del sistema

3. Formación del usuario

3.1. Manuales y Cursos

Aprendizaje y formación	Unidades	Precio
Manual de usuario TCL 8.6	1	50,95 €
Curso de introducción a TCL (4h)	1	89 €
	TOTAL:	139,95 €

Tabla 18. Sumas parciales correspondientes al material formativo para futuros usuarios

Capítulo 4

Presupuesto general

Esta última parte del presupuesto recoge la suma de todos los costes unitarios del proyecto en una única tabla, obteniéndose así los costes totales del proyecto:

1. Costes totales del proyecto

Concepto		Precio
Dispositivos de medición		12.749,00 €
Dispositivos de control		550,00 €
Otros		666,66 €
Software		0,00 €
Diseño de librerías		13.965,66 €
Aprendizaje y formación		139,95 €
	TOTAL:	**28.071,27 €**

Tabla 19. Sumas parciales correspondientes a los dispositivos de medición